Nr 1 kwiecień 1986 Nr ineksu 36-345 Cena 100 zł ISNN 0860-2541

KOMPUTER

POPULARNY MIESIĘCZNIK INFORMACYJNY



10 REM

„Nie ma jutra bez komputra" – to modne wśród informatyków hasło dodawało nam otuchy podczas wielomiesięcznych starań o utworzenie samodzielnego, popularnego pisma informatycznego. Mieszczą się w nich między innymi dzieje tworzonego z tą myślą „Bajtka", pisma będącego ostatecznie – po przekazaniu tytułu do redakcji Sztandaru Młodych – dodatkiem do tej gazety.

Na kombatanckie wspominki przyjdzie jednak czas, gdy dane nam będzie świętować kolejne rocznice. Dziś, a piszemy te słowa w dniu zakończenia XXVII Zjazdu KPZR, chętniej patrzymy w przyszłość. Nasza ufność, że niesie ona nam wiele przygód intelektualnych i technicznych, opiera się nie tylko na słowach Uchwał i Programów, ale i na towarzyszącym im klimacie.

Ten klimat towarzyszy i wyzwaniu lat osiemdziesiątych – komputerom osobistym. Obok agitacji („komputer wielką rzeczą jest i basta!") i dywagacji udaje się znaleźć miejsce dla rzetelnej intormacji i edukacji technicznej. Przykład zresztą idzie z góry. O przyspieszeniu świadczą m.in. pierwsze komunikaty z posiedzeń Biura Politycznego KPZR po wyborze nowego przywódcy – w ciągu miesiąca m.in. dwukrotnie rozpatrywano sprawę edukacji i upowszechnienia kultury informatycznej, formułując sprawdzalne, operacyjne cele i zadania. Ten ton utrzymano w dyskusji przedzjazdowej i podczas Zjazdu. Sprostanie wyzwaniu najnowszych technologii i odzyskanie czołowych pozycji w tym zakresie nie jest dziś jednym z wielu, lecz podstawowym celem Partii Lenina. To zaś, że potrafi ona skutecznie realizować swe podstawowe zadania w najtrudniejszych warunkach, udowodniła już nieraz.

Podobny ton można odnaleźć w dokumentach programowych publikowanych przed X Zjazdem PZPR, którego uchwały, jak sądzimy, przyspieszą rozwój polskiej „dużej" i „małej" informatyki. Podobny styl widać też w działaniach Urzędu ds. Nauki i Postępu Technicznego, choć skuteczność tych działań nie zawsze jeszcze jest na miarę oczekiwań i możliwości.

Mamy nadzieję, że "KOMPUTER" dobrze się przyczyni do realizacji tych starych, ale na nowo sformułowanych zadań. Naszym głównym celem jest szerzenie kultury informatycznej wśród dojrzałych Czytelników, stających wobec konieczności posługiwania się środkami informatyki i rozumienia ich. Na świecie cel ten określany bywa jako „computer literacy" lub „kompjuternaja gramotnost'", my jednak zamiast o dziwacznie w języku polskim brzmiącej „alfabetyzacji" wolimy mówić o kulturze informatycznej.

Chcielibyśmy więc, by "KOMPUTER" okazał się przydatny dla nauczycieli, zarówno tych uczących informatyki, jak i pozostałych; dla instruktorów klubów mikrokomputerowych, dla obecnych i potencjalnych użytkowników sprzętu i systemów informatycznych; dla polityków i innych osób zainteresowanych społecznymi skutkami upowszechnienia informatyki. Przewidzieliśmy oczywiście i coś lżejszego, przecież chyba każdy z nas po skończonej pracy lubi pograć w gry zgromadzone przez swoje dziecko.

Mamy nadzieję, że wszystkich naszych Czytelników zachęcimy do traktowania komputera nie jako przedmiotu zachwytów, źródła prestiżu lub modnej zabawki, lecz jako normalnego przedmiotu użytkowego, który właściwie wykorzystany już dziś jest w stanie udowodnić swą przydatność prawie w każdym miejscu pracy i w wielu domach. Aby jednak mógł on spełnić takie zadanie, użytkownik, poza życzliwą dlań postawą, musi mieć jeszcze wiedzę. Będziemy się starali dostarczyć ją Wam, Drodzy Czytelnicy, w możliwie łatwo przyswajalnej formie.

Pierwszy numer naszego pisma, który leży przed Wami, daleki jest jeszcze od naszych o nim wyobrażeń. W części jest to cena, jaką płacimy za pośpiech; od uzyskania najważniejszej z pozytywnych decyzji do momentu ukazania się pisma w kioskach minie

SPIS TREŚCI ■ SPIS TREŚCI ■ SPIS TREŚCI ■ SPIS TREŚCI ■ SPIS TR

MENU:



mniej niż miesiąc. Dla miesięcznika jest to tempo godne uwagi, zwłaszcza że trzeba było pokonać liczne trudności techniczne, rozwiązać wszystkie sprawy biurokratyczno-papierkowe, a wszystko to w zespole znacznie skromniejszym od zakładanego na przyszłość. Udało nam się dzięki pomocy, często pracochłonnej a bezinteresownej, kilkudziesięciu współpracowników, którym chcemy tu gorąco podziękować za ich pracę.

Mamy nadzieję, że przy realizacji dalszych naszych planów będziemy mogli liczyć na Waszą pomoc i współudział. Pracować będzie nad czym: poza redagowaniem miesięcznika zamierzamy w najbliższym czasie rozpocząć seryjną produkcję kaset z programami, których ceny będą konkurencyjne w stosunku do oferty z bazaru. Szczególnie gorąco zachęcamy do nawiązania z nami kontaktu, autorów oryginalnych, powstałych w Polsce programów – zamierzamy im zaproponować godziwy procent wpływów ze sprzedaży ich dzieł. Sądzimy, że dla wielu użytkowników mikrokomputerów biblioteka programów "KOMPUTERA", sprawdzonych, profesjonalnie nagranych i wyposażonych w starannie opracowane instrukcje będzie atrakcyjną propozycją. Specjalną serię kaset wydawać będziemy z myślą o szkołach, we współpracy z Ministerstwem Oświaty i Instytutem Badań Pedagogicznych.

Już wkrótce ukażą się także w kioskach pierwsze broszury z serii biblioteczki "ABC – KOMPUTERA". Zawierać one będą porady dla użytkowników poszczególnych typów mikrokomputerów, kursy języków programowania, informacje o sprzęcie i o specjalnych jego zastosowaniach różnego rodzaju.

W samym piśmie, obok form i tematów obecnych w pierwszym numerze, planujemy to wszystko, co Was zainteresuje, Drodzy Czytelnicy i co może być przydatne dla tych wszystkich, którzy pierwszy kontakt z komputerem mają już za sobą.

Mamy nadzieję, że naszym wspólnym hasłem będzie:

„MYŚL SAM,
KOMPUTER CI POMOŻE!"

Warszawa, 1986 r. REDAKCJA

KOMPUTER

KOMPUTER NR 1
KWIECIEŃ 1986

Popularny Miesięcznik Informatyczny pismo miłośników i użytkowników mikrokomputerów redagują:
Marek Młynarski (red. nacz.)
Władysław Majewski (z-ca red. nacz.)
Elżbieta Bobrowska (z-ca sekr. red.)
oraz: Andrzej Bączyński, Rafał Brzeski, Marek Car, Grzegorz Czapkiewicz, Andrzej Kadlof, Jarosław Kania, Zbigniew Kasprzycki, Krzysztof Kuryłowicz, Jacek A. Likowski, Leszek Rudak, Grzegorz Szewczyk, Jakub Tatarkiewicz, Darosław J. Toruń, Piotr Tymochowicz, Roland Wacławek, Tadeusz Wilczek, Tomasz Zieliński
opracowanie graficzne: Beata Maruszewska
redaktor techniczny: Anna Szubka
wydawca: Krajowe Wydawnictwo Czasopism RSW „Prasa-Książka-Ruch", ul. Noakowskiego 14, 00-666 Warszawa, tel. centr. 25-72-91 do 93
redakcja: ul. Mokotowska 48, 00-543 Warszawa, tel. 21-76-58 telex 815664 cestud pl (budynek Warszawskiego Centrum Studenckiego Ruchu Naukowego ZSP, również wspierającego pismo).
Skład, druk i opracowanie: Prasowe Zakłady Graficzne, Łódź, ul. Armii Czerwonej 28. Zam. 704/86
cena egzemplarza 100 zł
prenumerata: kwartalna – 300 zł, półroczna – 600 zł, roczna – 1200 zł.
Instytucje i zakłady pracy zamawiają prenumeratę w oddziałach RSW „Prasa-Książka-Ruch" lub urzędach pocztowych, jeśli w ich miejscowości nie ma oddziału RSW. Prenumeratę indywidualną przyjmują urzędy pocztowe, a na wsi także doręczyciele, na „blankietach wpłaty" na rachunek miejscowego oddziału RSW.
Prenumerata ze zleceniem wysyłki za granicę jest o 50% droższa dla osób prywatnych i o 100% dla instytucji, a przyjmuje ją Centrala Kolportażu RSW, ul. Towarowa 28, 00-958 Warszawa, konto NBP XV O/M W-wa 1153-201045-139-11.
Prenumerata na rok następny przyjmowana jest do 10 listopada, a na II, III i IV kwartał i II półrocze – z miesięcznym wyprzedzeniem.
Ogłoszenia przyjmuje Biuro Reklamy, ul. Mokotowska 5, tel. 25-35-36 (korespondencję prosimy kierować pod adresem Wydawnictwa: Noakowskiego 14, 00-666 Warszawa).
Opłaty za ogłoszenia (od osób prywatnych z góry) wpłacać można na konto KWCz w NBP III O/M W-Wa 1036-5294 zaznaczając na blankiecie „ogłoszenie w KOMPUTERZE" lub w kasie Wydawnictwa.
1 cm² ogłoszenia kosztuje 300 zł, najmniejsze ogłoszenie – 2100 zł (ok. 20 słów); kolor dodatkowy – 30% drożej, pełna gama barw – 100% drożej.
Ogłoszenia można zamawiać listownie, podając datę i miejsce wpłaty.
Nakład 150.000. P-93.
Nr indeksu 36-345 ISSN 0860-2541

 EŚCI ■ SPIS TREŚCI ■ SPIS TREŚCI ■

ZAMIAST PROGRAMU

W tym miejscu ukazać miał się artykuł o rządowym programie edukacji informatycznej. Zgodnie z Uchwałą Prezydium Rządu z listopada 1984 r. powinien on być uchwalony najpóźniej w październiku 1985 r.

Projekty gotowe są od wielu miesięcy. Ich pierwsze wersje opracowane zostały w lipcu ub. roku.

W październiku 1985 r. kolegium Ministerstwa Oświaty zatwierdziło projekt programu, który na początku grudnia przesłany został do konsultacji międzyresortowych. Do projektu zastrzeżenia zgłosili minister finansów i przewodniczący Komisji Planowania oraz minister ds. młodzieży, który uznał, że zbyt małą rolę powierzono w nim klubom pozaszkolnym.

Obecnie trwają prace nad wspólnym projektem trzech resortów: oświaty, szkolnictwa wyższego i młodzieży.

***

Zamiast artykułu o programie edukacji mógłby ukazać się tu materiał o przygotowywanym przez Urząd ds. Nauki i Postępu Technicznego zamówieniu rządowym na mikrokomputer szkolny. W tej sprawie korespondencja między wrocławskimi Zakładami Elektronicznymi ELWRO a Urzędem i Ministerstwem Oświaty trwa od czerwca ub. roku. W lipcu ub. r. w Miedzeszynie k. Warszawy odbyło się spotkanie, podczas którego uzgodniono wszystkie warunki takiego zamówienia. Od tego czasu wszystkie zainteresowane instytucje obciążają się winą za przedłużanie się konsultacji.

Częścią zamówienia rządowego ma być program badawczy w zakresie oprogramowania dydaktycznego. Do czasu zatwierdzenia zamówienia odpowiednie instytucje badawcze i resorty nie mogą wydać nawet kilku tysięcy złotych na ten cel. Obecnie nawet sieć lokalna MERITUM wraz z jej wcale już bogatym oprogramowaniem tworzona jest praktycznie na własny koszt ośrodka obliczeniowego Politechniki Śląskiej.

Brak decyzji w sprawie zamówienia oznacza, że praktycznie do tej chwili nie wiadomo, jaki będzie przyszły mikrokomputer szkolny. Skutecznie powstrzymuje to prace nad jego oprogramowaniem, gdyż mało kto ma ochotę inwestować wielomiesięczny wysiłek w coś, co może okazać się tylko przejściową gwiazdą kilku sezonów.

Brak decyzji w tej sprawie oznacza również, że Zakłady Elektroniczne ELWRO zamiast dziesięciu tysięcy sztuk modelu docelowego wyprodukują w tym roku zaledwie kilkaset lub niewiele ponad 1000 sztuk MERITUM (którego produkcję będą stopniowo przejmować z ELZAB-u) oraz co najwyżej kilkaset sztuk serii próbnej właściwego mikrokomputera dla szkół.

***

Podejmowane obecnie decyzje będą miały wpływ na kształt materialny i organizacyjny edukacji informatycznej i ruchu mikrokomputerowego w ciągu wielu najbliższych lat. Dobrze więc, że przygotowywane są starannie i rozważnie, a opinia publiczna informowana jest o pojawiających się propozycjach i jej głos jest uważnie wysłuchiwany.

Nawet bardzo starannie przygotowywana decyzja musi jednak kiedyś zostać podjęta. W trakcie wielomiesięcznego już snu zimowego i okresu ogólnej niemożliwości świat nie stał w miejscu. Ceny mikrokomputerów spadły o kolejne 20-30%, standardem stało się 128 K pamięci dla najprostszych zabawek domowych oraz 1 MB dla mikrokomputerów osobistych (IBM PC AT, Macintosh+, Atari 1024, Commodore Amiga) – a urządzenia te można kupić już za około 1000 dolarów. Są to pojemności jeszcze przed rokiem, gdy formułowano założenia z takim trudem rodzących się zamówień i programów, niewyobrażalne – warto przypomnieć, że jeszcze latem 1985 r. zupełnie poważnie na najwyższym szczeblu rządowym rozważano zakup dla oświaty 100 tys. szt. komputereczków ZX 81 z pamięcią... 1 KB.

Nie spano również w kraju. Nie czekając na decyzje rządowe utworzono Fundację Mikrokomputerową, choć jej rejestracja również przeciągała się przez wiele miesięcy. Wiele instytucji fundowało szkołom sprzęt, kupując to, co akurat można było kupić. Mamy więc dziś w szkołach Commodory (Poznań), ZX Spectrum, portugalskie Timex-y z naklejką UNIPOLBRIT i Timex-y z Baltony, Atari 800 XL z Pewexu.

Nawet samo Ministerstwo Oświaty nie mając innego wyboru postanowiło znaczne kwoty zarezerwowane na wyposażenie ośrodków kształcenia nauczycieli przeznaczyć na zakup ZX Spectrum. Kształcenie nauczycieli nie może przecież czekać. W efekcie z całą pewnością przez najbliższych kilka lat dla przeszkolonego nauczyciela synonimem słowa mikrokomputer będzie ZX Spectrum.

Po fiasku zarówno oddolnych, jak i odgórnych prób utworzenia Federacji Klubów Mikrokomputerowych obejmującej wszystkich zainteresowanych, Kluby Turnieju Młodych Mistrzów Techniki ZSMP utworzyły własną Federację pod patronatem Urzędu ds. Postępu Technicznego.

***

Realne staje się więc niebezpieczeństwo, że gdy wiosną lub latem Rząd uchwali już najstaranniej przemyślany i uzgodniony Program – niewiele będzie do programowania i do regulowania. Siła okrzepniętych już faktów dokonanych pokona każdego Herkulesa. Przysłowie mówi: Dwa razy daje, kto szybko daje.

Danuta i Władysław Majewscy

W imieniu Instytutu Badań Pedagogicznych i Departamentu Organizacji Badań i Informacji Pedagogicznej zwracamy się do wszystkich zajmujących się opracowywaniem oprogramowania dydaktycznego, o skontaktowanie się z Pracownią Zastosowań Informatyki Instytutu Badań Pedagogicznych, Warszawa, ul. Górczewska 8.

Pracownia prowadzi katalog dostępnego w naszym kraju oprogramowania dydaktycznego. Szersze informacje na ten temat znajdą się w następnych numerach "KOMPUTERA".

Równocześnie informujemy, że w dniach od 18 do 20 czerwca 1986 r. odbędzie się w Wałbrzychu II Krajowa Konferencja „Informatyka w szkole".

Zgłoszenia przyjmuje Rada Wojewódzka NOT w Wałbrzychu, ul. Szmidta 4a, 58-300 Wałbrzych, tel. 235-69.

Redakcja "KOMPUTERA" prosi równocześnie organizatorów wszelkich konferencji, seminariów i otwartych kursów o tematyce związanej z zastosowaniami informatyki i wykorzystaniem mikrokomputerów o nadsyłanie nam informacji o terminach, tematach i warunkach uczestnictwa z co najmniej 3 miesięcznym wyprzedzeniem. Planujemy, począwszy od następnego numeru, stworzenie specjalnej rubryki poświęconej tego rodzaju informacjom oraz omówieniom imprez, na które zostaliśmy zaproszeni.

Informacje o imprezach, w których uczestnictwo jest bezpłatne i otwarte, zamieszczamy bezpłatnie, a o płatnych i zamkniętych – w razie zaproszenia przedstawiciela redakcji i wyrażenia zgody na publikację informacji o wynikach imprezy oraz wygłoszonych referatach.

WPROWADZENIE

Większość użytkowników i kibiców mikrokomputerów trwa w przekonaniu iż najważniejszą sprawą jest sprzęt, a o jakości systemu komputerowego decydują zawarte w nim kości. Przekonanie to jest o tyle usprawiedliwione, że korzystając ze Spectrum istotnie mamy do wyboru tylko jeden, firmowy system operacyjny. Podczas prelekcji i wykładów spotykamy się z opiniami brzmiącymi w uszach specjalisty absurdalnie, np: "przecież wszystkie komputery osobiste mają wbudowany BASIC!". Wypowiadający te słowa nie chcą zazwyczaj uwierzyć, iż większość używanych na świecie komputerów osobistych typu IBM PC i podobnych nie ma na stałe wbudowanego żadnego języka.

Dla wielu ludzi dziwacznie brzmi spotykane często zdanie, iż dany program "chodzi pod CP/M". W kilku najbliższych numerach spróbujemy Naszym Czytelnikom wyjaśnić znaczenie takich uwag oraz przekazać podstawowe informacje dotyczące najbardziej rozpowszechnionych systemów operacyjnych. Rozpoczynamy od omówienia systemów CP/M i UNIX. W dalszych numerach dokończymy opis CP/M i przedstawimy CP/M 86 i MS DOS wraz z jego kolejnymi wersjami.

Pojęcie "system operacyjny" w zasadzie oznacza ogół bloków programowych, umożliwiających komputerowi podjęcie dialogu z użytkownikiem i zarządzanie swoimi zasobami. W tym sensie swego rodzaju systemem operacyjnym dysponuje także ZX Spectrum. W praktyce jednak utarło się pewne węższe rozumienie tego pojęcia. Posługując się nim myślimy zazwyczaj o programach zarządzających systemem dyskowym. Świadczą często o tym same nazwy, np. Disc Operating System, które pozwalają oprogramowaniu użytkowemu współpracować bezpośrednio z dyskiem i to niezależnie od szczegółów organizacji współpracy komputera i dysku.

Artykuł o CP/M jest przedrukiem z Biuletynu Polskiego Towarzystwa Informatycznego, znakomitego miesięcznika, ukazującego się niestety w nakładzie tylko 1000 egz. Redakcji i Zarządowi Głównemu PTI dziękujemy za zgodę na korzystanie z materiałów Biuletynu.

CP/M (część pierwsza)

Historia systemu CP/M (Control Program for Microcomputer) sięga już 11 lat. W roku 1974 Kanadyjczyk Gary Kildall stworzył pierwszą wersję CP/M dla swego mikrokomputera IMSAI. Istniały wprawdzie wcześniejsze rozwiązania monitorów dotyczące mikrokomputerów z pamięcią masową w postaci dysków elastycznych, były to jednak czasy, w których bezsprzecznie królowała taśma papierowa. Niemniej jednak przedsięwzięcie Kildalla – stworzenie systemu operacyjnego dla mikrokomputera z pamięcią masową o organizacji plikowej, z definiowanymi strumieniami wejścia i wyjścia, przy pamięci operacyjnej ok. 16 KB – było epokowym wydarzeniem w historii informatyki.

W chwili obecnej system operacyjny CP/M jest standardem de facto, jeśli chodzi o mikrokomputery ośmiobitowe z pamięcią masową w postaci dysków elastycznych. Kolejno na rynku oprogramowania pojawiały się coraz nowsze wersje tego systemu: 1.3, 1.4, 2.0, 2.1 i 2.2. Stało się to za sprawą samego Kildalla i jego współpracowników z firmy Digital Research (Pacific Grove California). Ale prawdziwy sukces handlowy zawdzięcza CP/M przedsiębiorczemu Tony'emu Goldowi, który założył w Nowym Jorku firmę handlową (software house) Lifeboat Associates. Firma ta zajmuje się głównie dystrybucją systemu CP/M i programów pracujących pod jego nadzorem. Warto dodać, że biblioteka programów zgodnych z systemem CP/M, z wielu różnych firm, sięga 3 tys. pozycji. Bardzo istotnym elementem sukcesu systemu CP/M był fakt, że napisany początkowo na MDS 800, dzięki umiejętnym modyfikacjom, stał się systemem najłatwiej przenoszonym i instalowanym na różnych mikrokomputerach. Pracuje na mikroprocesorach 80 80, 80 85 i Z 80. Istotną, aczkolwiek mającą obecnie coraz mniejsze znaczenie zaletą jest także niewielka wymagana pamięć operacyjna (16 KB dla wersji 1.4 i 20 KB dla wersji 2.0 i późniejszych). Łatwość przystosowania programu do środowiska sprzętowego, konfiguracji, typu pamięci dyskowej (dyski elastyczne lub sztywne) oraz niekorzystanie z przerwań – były to wszystko przyczynki do sukcesu, w którym połączono dobry produkt z umiejętnym marketingiem.

Charakterystyka systemu CP/M

Ograniczymy się tutaj do wersji 2.2, która z jednej strony stanowi produkt w pełni dojrzały, z drugiej – osiągalny za niewielką sumę, bo ok. 120 dolarów, a na dodatek dostępny mikroinformatykom w Polsce.

System CP/M 2.2 jest uniwersalnym, jednostanowiskowym systemem operacyjnym dla mikrokomputerów wyposażonych w mikroprocesor Intel 80 80 lub zgodne z nim mikroprocesory 80 85 i Z 80 oraz pamięć masową na dyskach elastycznych pojedynczej gęstości – standard IBM 3740 lub podwójnej gęstości – standard IBM 34, 8 lub 5.25". Wymaga ciągłego bloku pamięci RAM o wielkości minimalnej 20 KB począwszy od adresu 0. Oprogramowanie napisane i działające pod nadzorem systemu CP/M daje się w pełni przenieść na dowolną maszynę wyposażoną w ten sam system operacyjny. Blok programów komunikacji z pamięcią masową jest oparty na tzw. tablicach parametrów dyskowych, umożliwiających łatwą zmianę konfiguracji zasobów sprzętowych mikrokomputera. Informacja na dyskietkach ma postać plików o unikalnych nazwach.

Oryginalne oprogramowanie dostarczane przez Digital Research lub innego autoryzowanego dystrybutora jest przekazywane odbiorcy na dysku elastycznym wraz z 7 tomami dokumentacji opisującej poszczególne

fragmenty systemu, ich przeznaczenie i sposób użytkowania oraz programy usługowe, tzw. utilities.

System CP/M jest systemem jednoprogramowym, z możliwością realizacji wieloprocesowości. Jest to system na tyle dobrze zaprojektowany, że użytkownik może zapomnieć o fizycznym istnieniu dysków. Wydaje się jednak celowe poznanie architektury i sposobu pracy systemu, gdyż takie podejście zawsze pozwoli racjonalniej użytkować dostępne zasoby, ułatwi zrozumienie reakcji systemu na poszczególne sytuacje i w efekcie da możliwość wyzyskania wszystkich zalet tego systemu operacyjnego.

Budowa systemu

System operacyjny CP/M 2.2 dzieli się logicznie na 5 części. Podział ten wynika ze struktury pamięci operacyjnej, a właściwie ze sposobu wykorzystania poszczególnych jej obszarów i rezydujących w nich programów.

Stronica zerowa (Page Zero, PZ)

Obszar zapewniający łączność między rezydującą częścią systemu i programami użytkownika. Zawiera skoki do programu ładującego i do modułu bazowego. Oprócz tego są w nim umieszczone niektóre pola robocze oraz bufor dla operacji dyskowych i komunikacji z konsolą. Stronica zerowa jest generowana podczas startu lub ponownego startu systemu przez program z modułu BIOS.

Obszar programów nie rezydujących (Transient Program Area, TPA)

Obszar pamięci, począwszy od adresu 100H, do którego są wprowadzane z dyskietek programy użytkownika. W tym obszarze są także wykonywane tzw. polecenia systemowe.

Procesor komend operatorskich (Console Command Processor, CCP)

Obszar, do którego jest przepisywany – podczas wprowadzania systemu z dyskietki – program obsługujący komunikację z operatorem oraz wykonujący rezydentne polecenia systemowe. Ten program wprowadza do obszaru TPA wywołane polecenia nierezydentne i programy użytkownika.

Bazowy system operacyjny (Basic Disc Operating System, BDOS)

Jądro systemu operacyjnego. Jest to zestaw programów wprowadzanych z dyskietki podczas inicjalizacji systemu, zawierający zbiór procedur obsługujących pliki, obsługujących 1 do 16 jednostek pamięci na dyskach elastycznych, znakowe strumienie wejścia i wyjścia oraz zarządzanie pamięcią RAM. Jest to obszar, który nie może być niszczony przez programy użytkowe.

Bazowy system wejścia/wyjścia Basic I (O System, BIOS)

Zbiór programów i tablic zapewniających fizyczną komunikację ze sprzętem w danej instalacji. Jest to jedyny moduł systemu CP/M, który jest zależny od środowiska sprzętowego i musi być wygenerowany dla danego mikrokomputera indywidualnie. Wprowadza się go z dyskietki podczas inicjalizacji, aczkolwiek zawiera on także część będącą zazwyczaj w pamięci stałej typu ROM (programy obsługi we/wy). Moduł ten nie może być niszczony przez programy pracujące pod kontrolą systemu CP/M.

Poszczególne części systemu CP/M zajmują kolejne, przyległe obszary pamięci operacyjnej mikrokomputera, a więc pamięć, która znajduje się powyżej obszaru modułu BIOS, nie jest przez system używana i może mieć dowolne przeznaczenie. Jedynie stronica zerowa BDOS i BIOS są elementami systemu CP/M, które muszą pozostawać w pamięci przez cały czas pracy systemu. Obszar zajmowany przez moduł CCP może być nakładkowany, jeśli to okaże się potrzebne. W takim przypadku powrót do systemu po wykonaniu programu musi być poprzedzony ponownym startem CP/M, tzw. gorącym startem.



Ciąg dalszy w następnym numerze.

Andrzej J. Majewski
Instytut Informatyki
Politechniki Gdańskiej

DLACZEGO WŁAŚNIE UNIX?

System UNIX stał się modny. Każdy, kto ma jakąś nieprzypadkową styczność z informatyką, zna tę nazwę. Ale czy to UNIX jest taki popularny, czy jego nazwa?

Aby odpowiedzieć na to pytanie, należy pokrótce opisać obiekt zainteresowania. Można powiedzieć, że jest to system operacyjny (*) dla maszyn rodziny PDP-11, przeniesiony na szereg innych komputerów. Jest to system wielodostępny (*), wielozadaniowy (*) (poprzez podział czasu procesora (*)), z hierarchicznym systemem zbiorów (*). Ale takie wyliczanie cech, bardziej i jeszcze bardziej szczegółowe, nie tłumaczy popularności systemu UNIX. O taką popularność zabiegali autorzy wielu systemów, przekonani o ogromnych walorach swoich produktów, a "życie" bardzo brutalnie zweryfikowało ich mniemania. Autorzy UNIX wyróżniali się tym, że nie byli projektantami systemów operacyjnych, lecz użytkownikami. Ich naczelną zasadą było: system operacyjny powinien być prosty i łatwy w użyciu. Ta idea doprowadziła do powstania systemu atrakcyjnego dla użytkowników. A to przecież oni w ostatecznym rozrachunku oceniają system, nie zaś koledzy po fachu panów projektantów. A więc podstawą popularności systemu stał się fakt, że został zrobiony przez użytkowników i dla użytkowników. Powstało wiele systemów operacyjnych zgodnych funkcjonalnie z UNIX (np. Xenix, dla IBM PC, Tunis). Ale na tym nie koniec. Rozwiązania przyjęte w systemie UNIX zostały użyte przy budowie systemów różnych funkcjonalnie od pierwowzoru. Charakterystyczne, że projektanci systemów przyjmują elementy UNIX jako rozwiązania "naturalne", znane "od zarania". Bywa, że nie są świadomi zapożyczeń.

Informacja jest zorganizowana w systemie UNIX w system zbiorów o strukturze hierarchicznej. Składnikami systemu są zbiory zwyczajne i katalogi. Katalog zawiera informacje o zbio-



rach podległych mu w hierarchii (w tym również katalogach). Pracujący użytkownik ma przypisany "aktualny katalog". Istnieje mechanizm zmiany aktualnego katalogu. Do zbioru można się odwoływać przez podanie jego nazwy, jeżeli jest to zbiór lokalny w aktualnym katalogu, lub przez podanie prowadzącej do niego "ścieżki" (*) kolejnych katalogów. Prosty mechanizm ochrony pozwala wyłączyć użytkownika bądź grupę użytkowników z dostępu do zbioru.

Informacja w zbiorze jest uporządkowanym ciągiem bajtów, bez rozróżniania fizycznych czy logicznych jednostek informacji innych niż bajt. W każdej chwili dostępny jest dowolny bajt zbioru.

Urządzenia zewnętrzne są traktowane w systemie jak zbiory. Transformacji informacji z ciągu bajtów na postać specyficzną dla urządzenia dokonuje system, a użytkownik nie musi nic wiedzieć o specyfice urządzenia.

Wywołanie programu polega na napisaniu nazwy zbioru, w którym program jest przechowywany. Parametry (*) są przekazywane programowi jako ciągi znaków oddzielone odstępami, umieszczone za nazwą programu. Jeżeli parametrem jest grupa zbiorów o zbliżonych nazwach, specjalna notacja pozwala na symboliczną kompresję ich do jednej nazwy.

Programy mają pewne standardowe "strumienie" (*) wejścia/wyjścia, normalnie związane z terminalem (*), z którego użytkownik prowadzi konwersację z systemem. W łatwy sposób można te strumienie związać z dowolnym zbiorem.

Bardzo istotną cechą systemu UNIX jest możliwość łączenia wyjściowego strumienia informacji jednego programu z wejściowym strumieniem drugiego (mechanizm ten jest znany pod nazwą **pipe)**. Pozwala to uniknąć tworzenia zbiorów pośrednich, przechowujących informacje wyprodukowane przez pierwszy z programów.

Wymienione powyżej cechy dotyczące uruchamiania programów są cechami interpretera zleceń (*) użytkownika w systemie UNIX, zwanego SHELL. SHELL nie jest jednak specjalnie wyróżnionym programem i użytkownik może go zastąpić dowolnym innym programem. Jest to cecha ogólna, to znaczy wszystkie programy systemowe są "systemowe" tylko dlatego, że są umieszczone w wyróżnionym katalogu. Poza tym niczym nie różnią się od innych programów i użytkownik może je zastąpić własnymi.

Program SHELL dostarcza bardzo wygodnego języka makrozleceń (*), przy pomocy którego można tworzyć programy przez połączenie kilku-kilkunastu programów (zwanych przez autorów systemu narzędziami programowymi).

Wymienione powyżej cechy czynią z UNIX system "otwarty", nie posiadający ustalonej liczby i postaci dyrektyw oraz ustalonego sposobu konwersacji. UNIX był pierwszym systemem tego typu. Ta otwartość pozwoliła na stworzenie środowiska, będącego zarówno silnym narzędziem dla doświadczonych programistów, jak i bardzo wygodnym dla początkujących.

Wreszcie ostatnia z cech, które chcę wymienić (co nie znaczy, że najmniej ważna), to język C. Jest to język programowania powszechnie używany w systemie UNIX do produkcji owych "narzędzi programowych", a również do pisania całkiem dużych programów, jak kompilatory języków. Sam UNIX został napisany w języku C, co pozwoliło na łatwe przenoszenie systemu na inne komputery. O sile tego języka decyduje fakt połączenia w nim cech języków strukturalnych (ALGOL, PASCAL, PL/1) z językami symbolicznymi (asemblery).

Czytelnik może zapytać: co jest w tych cechach systemu UNIX rewelacyjnego? Przecież są to zwyczajne cechy, które można znaleźć w tak popularnych systemach jak CP/M czy MS-DOS. Ano właśnie, teraz można. Ale jeszcze przed dziesięciu laty właśnie system UNIX był nowością. Projektantom systemów operacyjnych do głowy nie przychodziło, że tak prosty system (niektórzy mówili złośliwie: prostacki) może zyskać uznanie użytkowników. A to, że cechy UNIX są teraz "zwyczajne", świadczyć może tylko na jego korzyść.

Wiktor B. Daszczuk

Słownik użytych terminów

system operacyjny – program pośredniczący między komputerem rozumianym jako sprzęt a użytkownikiem i jego programami

interpreter zleceń – część systemu operacyjnego odpowiedzialna za konwersację z użytkownikiem

wielodostępność – możliwość pracy z komputerem przez wielu użytkowników jednocześnie

wielozadaniowość – możliwość uruchomienia przez użytkownika więcej niż jednego programu (zadania) jednocześnie

podział czasu procesora – przydzielenie poszczególnym zadaniom małych odcinków pracy procesora, tak że użytkownik ma wrażenie, jakby każde zadanie miało przydzielony oddzielny procesor

ścieżka – sposób poruszania się w systemie zbiorów; np. aby osiągnąć zbiór f, można podać ścieżkę ACAf, lub, jeżeli aktualnym katalogiem jest C, ścieżkę Af

parametry programu – informacje zmieniające sposób pracy programu; np. dla wywołania programu kopiuj kopiującego zbiór do innego zbioru:
kopiuj a b l

parametry mogą znaczyć:

a – zbiór kopiowany

b – zbiór, do którego ma zostać skopiowany zbiór a

l – informacja o tym, że małe litery zbioru a mają być w czasie kopiowania zamienione na duże

(oczywiście znaczenie parametrów zależy od sposobu napisania programu kopiuj)

strumień – połączenie między źródłem a ujściem informacji, np. między programami, między programem a terminalem itp.

terminal – urządzenie (zazwyczaj monitor alfanumeryczny), z którego użytkownik prowadzi konwersację z komputerem

makrozlecenie – zespół często wykorzystywanych zleceń systemu operacyjnego, używanych w typowych połączeniach

SPECTRUM
UCZY I BAWI

Pomysł zastosowania komputerów w procesie nauczania nie jest nowy, lecz dopiero wprowadzenie na rynek setek tysięcy bardzo tanich (jak na komputery i na warunki zachodnie) maszyn stworzyło realne możliwości jego realizacji na szeroką skalę. Wydawcy oprogramowania oferują odpowiednie programy edukacyjne, które są nie mniej popularne, co gry.

Twój nauczyciel

Są to najczęściej lekcje komputerowe, w których komputer, jak nauczyciel ,prezentuje pewną partię materiału. Na ekranie ukazuje się treść wykładu, rysunki, tłumaczenia itd. Na zakończenie, albo też na bieżąco, sprawdza się stopień opanowania przez uczących się wiadomości, powtarza się trudniejsze fragmenty materiału, podsumowuje. Często spotyka się także komputerowe testy, sprawdziany i ćwiczenia, określone angielskim terminem "Drill And Practice".

Do nauczania stosuje się często symulatory, tzn. programy, które symulują przebieg różnych zjawisk fizycznych, czy procesów ekonomicznych i społecznych, wykorzystując do tego celu modele matematyczne.

Ważną cechą komputerowych programów edukacyjnych jest prowadzenie dialogu z uczniem. Komputer nie pozwala mu na bierność, zadaje pytania, reaguje na jego odpowiedzi, decyduje o dalszym przebiegu programu, ocenia stopień opanowania wiedzy, niekiedy nagradza ucznia. Nagrodą może być krótka zabawa z komputerem, innym razem pochwała wydrukowana na ekranie.

Programy edukacyjne dla mikrokomputerów ZX SPECTRUM zaadresowane są do różnych odbiorców, od przedszkolaków do dorosłych. I tak np. "Counting" ("rachunki"), to trzy programy dla dzieci, uczące liczenia do dwudziestu, a np. komputerowy kurs ASEMBLERA Z 80: "The Complete Machine Code Tutor" to blok programów uczących posługiwania się kodem maszynowym.

Firma SINCLAIR liczyła, że brytyjskie szkoły zakupią jej mikrokomputery ZX SPECTRUM. Jednak Anglicy zastosowali w szkołach wprawdzie droższe, lecz lepsze komputery BBC MICRO (jest ich teraz ponad 120 000). Niemniej ZX SPECTRUM używany jest do nauki w domach.

Między matematyką a literaturą

W Polsce jest używanych wiele angielskich programów edukacyjnych. Oto najpopularniejsze z nich:

Firma HOMESTUDY opracowała komputerowe kursy matematyki i fizyki z zakresu tzw. Secondary School – O Level (co w polskich szkołach odpowiada poziomowi klas od szóstej do dziesiątej). Skomputeryzowano dwa kursy matematyki według programu londyńskiego i "Cambridge" oraz program G nauczania fizyki. Na pojedynczej kasecie znajduje się kilkanaście programów edukacyjnych. Np. kurs matematyki według programu "Cambridge" (Cambridge Syllabus) ("Program podstawowy Cambridge"), składa się z 12 kaset. Obejmuje m. in. rachunek prawdopodobieństwa, ułamki, zbiory, funkcje, układy równań, macierze itd. Jedna z kaset zawiera program egzaminacyjny.

Wiele programów pomaga w nauce literatury angielskiej. Przykładem może być program firmy PENQUIN STUDY SOFTWARE pt. "Kupiec wenecki", omawiający sztukę Szekspira. Program zawiera streszczenie poszczególnych scen; można także zadawać komputerowi pytania dotyczące sztuki: napisanie wyrazu "Shylock" spowoduje wskazania w treści sztuki odpowiednich scen i dialogów związanych z tą postacią. Podobnym programem jest "Antoniusz i Kleopatra", "Hamlet" czy "Henryk IV" (także Szekspira) firmy AKADAMIAS SOFTWARE.

Zestawy programów firmy LONGMANS przeznaczone do nauczania fizyki i chemii ilustrowane są ruchomymi obrazami, czasami dość dowcipnymi i wesołymi, przyciągającymi więc uwagę ucznia. Programy te przypominają trochę filmy oświatowe. Opanowanie materiału ułatwiają zadania pomocnicze, rozwiązywane początkowo przez komputer, który pokazuje tryb rozumowania, a następnie przez ucznia, pod kontrolą komputera. Kolejne programy ilustrują trudniejsze partie materiału bądź podkreślają najbardziej istotne zagadnienia.

Dla najmłodszych

Najwięcej programów edukacyjnych przeznaczonych jest do nauczania początkowego, czytania oraz podstaw arytmetyki. Przykładem tego jest zestaw "Learn To Read" ("Nauka czytania") firmy MACMILLAN EDUCATION. Na początku dzieci uczą się wymawiania prostych słów. Trudniejsze słowa, wyświetlane na ekranie, ilustrowane są rysunkami. Następny program uczy dzieci układania prostych zdań. Uczeń opisuje określone zdarzenie uzupełniając w zdaniach brakujące wyrazy. Pozostałe programy uczą kolejności liter alfabetu oraz stosowania okoliczników miejsca (widocznie określanie miejsca sprawia angielskim dzieciom kłopoty i dlatego właśnie ten temat został wybrany). Nagrodą za dobre odpowiedzi jest zabawa, której czas zależy od liczby dobrych odpowiedzi. Dla przykładu w programie "Capital Letters" ("Duże litery"), który uczy zasad pisania dużą literą, zabawa polega na chwytaniu jabłek spadających z drzewa.

Przeżyć

Wśród programów edukacyjnych poczesne miejsce zajmują gry, które bawiąc – uczą. Przykładem może być program "Survival" ("Jak przeżyć"). Grający wybiera sobie zwierzę, np.

mysz, orła czy nawet lwa; każde zwierzę ma swych naturalnych wrogów, żywi się typowym pokarmem, lepiej lub gorzej znosi niską temperaturę, brak pożywienia i wody. Grający musi tak kierować swym zwierzakiem, by mógł on jak najdłużej przeżyć. Im dłużej dane zwierzę żyje, tym większe ma szanse na przetrwanie. Gra jest prosta i zawiera sporo wiedzy i ciekawostek o zwierzętach. Podobną grą jest "Sir Francis Drake" firmy LCL. Grający podróżuje razem ze sławnym żeglarzem – odkrywcą i korsarzem – dookoła świata, poznając geografię oraz fakty historyczne.

Przedstawione tu programy należą do bardzo "szkolnych". Są też i takie, które przeznacza się dla ludzi pragnących poszerzyć swoją wiedzę. Programy "How Machine Works ("Jak pracuje maszyna") firmy CALPAC COMPUTER SOFTWARE LTD opowiadają o zasadach działania samochodu i samolotu. Są one bogato ilustrowane ruchomymi, kolorowymi obrazami.

Kolejny program to "Astronomer" ("Astronom") marzenie wszystkich amatorów astronomii, czyli domowe planetarium. Pozwala on oglądać na ekranie telewizora niebo widoczne z każdego miejsca Ziemi o każdej porze roku i w dowolnie wybranej chwili. Program ten pokazuje również ruchomy model Układu Słonecznego.

Niektóre programy edukacyjne są wręcz zaskakujące. "Wheathermaster" ("Prognoza pogody") wydana przez MACMILLAN EDUCATION, uczy naukowych podstaw przepowiadania pogody, symulując zmiany pogody nad Atlantykiem.

Mały kompozytor

Dużym zainteresowaniem cieszą się programy uczące muzyki. Wykorzystują one możliwości wytwarzania przez komputer dźwięku. Np. Music Master uczy zasad zapisu nutowego, sposobu zapisywania nut, pauz, rytmu, melodii na pięciolinii. Uczeń ma możliwość wyboru jednego z kilku tematów, np. zasady podawania rytmu. Wykład wyświetlany na ekranie, ilustrowany jest dźwiękami. Uczeń uczy się zależności między dźwiękiem i nutą. Może porównać poszczególne dźwięki o różnej wysokości i różnym czasie trwania oraz zobaczyć jak wygląda ich obraz zapisany na pięciolinii.

Program daje także sposobność skomponowania własnej melodii, zapisanie jej w pamięci oraz na taśmie magnetofonowej a następnie odegranie w dowolnym tempie.

Chociaż program ten przeznaczony jest dla dzieci, wzbudza również zainteresowanie dorosłych.

Istnieją programy edukacyjne przeznaczone tylko dla dorosłych, a także dla starszej młodzieży. Przykładem może być wymieniony już zestaw uczący języka ASEMBLERA Z 80 – "The Complete Machine Code Tutor", lub "Beyond Basic" albo "Go Micro" (firmy LONGMANS), wyjaśniający sposób pracy i możliwości komputera.

Programy "Highway Code" ("Przepisy drogowe") firm DATELL COMPUTING, COMPUTER PENTALS i ROSE SOFTWARE, wyjaśniają przepisy i znaki drogowe oraz sprawdzają ich znajomość. Mogą być pomocne osobom przygotowującym się do egzaminów na prawo jazdy lub na kartę rowerową.

Disce puer latinae

Kolejną dziedziną, w której na świecie korzysta się z programów komputerowych, jest nauczanie języków obcych. Istnieją programy do nauki języka hiszpańskiego, niemieckiego, francuskiego i włoskiego. Do programów takich dołącza się często kasety zawierające dialogi i ćwiczenia fonetyczne. Programy komputerowe służą do ćwiczeń gramatyki. Zestawy firmy ROSE SOFTWARE "English 1" oraz "English 2" przeznaczone są do nauki języka angielskiego w brytyjskich szkołach dla uczniów 7-10-letnich. Programy te uczą synonimów, przysłów, wyrazów o znaczeniach przeciwnych, idiomów, ortografii itp. Ćwiczący uczeń wybiera jeden z kilku wydrukowanych na ekranie wyrazów, który stanowi odpowiedni synonim wyrazu brakującego w zdaniu. Uczeń wskazuje słowo podając jego numer.

Trzeba jednak przyznać, że niektóre programy są niewiele warte. Często firmy komputerowe dają swoim produktom etykietę programów edukacyjnych po to, by je lepiej sprzedać. Nie zawsze zastosowanie komputerów daje efekty proporcjonalne do nakładów. Często użycie komputera zamiast książki lub tablicy jest bezcelowe, a nawet niewskazane.



Krzysztof Kuryłowicz, Dariusz Madej, Krzysztof Marusek



ANUTA i WŁADYSŁAW MAJEWSCY

ZASADA PRZYWÓDCY

Profesor Gocza Czogawadze, doktor nauk technicznych w dziedzinie automatyki i techniki obliczeniowej, kierownik katedry Automatycznych Systemów Sterujących i główny konstruktor tych systemów Politechniki Gruzińskiej w Tbilisi. Od roku 1981 ekspert UNESCO do spraw edukacji informatycznej

– Panie Profesorze, wciąż dręczy nas pytanie, jak oceniać komputerowe programy dydaktyczne. Z pewnością UNESCO rozwiązało już ten problem?

– Nie umiałbym wskazać żadnego eksperta, który mógłby powiedzieć: to jest dobry program, właśnie takie programy należy robić. Do tej pory nie stworzono na świecie formalnych metod oceny programów. Większość podejmowanych prób ma charakter ekonomiczny. Porównywane są ceny programów i na tej podstawie wyciąga się nie zawsze słuszne wnioski. Dużo więcej moglibyśmy dowiedzieć się o programach, gdybyśmy wszystkie zgromadzone w Paryskim Centrum Dokumentacji Programów wypożyczyli do szkół, a po roku ich eksploatacji zapytali korzystających z nich nauczycieli: które z tych programów chcą pożyczyć znowu? Byłaby to miarodajna ocena programów, dokonana przez samych użytkowników, zależna zresztą od tego, kim oni są.

– A kim są użytkownicy?

– Możemy podzielić ich na cztery grupy. Pierwsza, stosunkowo wąska, to **zawodowi informatycy,** specjaliści w tej dziedzinie, absolwenci kierunków informatycznych uniwersytetów i wyższych szkół technicznych, biegli w programowaniu, językach wyższego rzędu, językach algorytmicznych, projektowaniu bez danych, systemów zarządzania bankami danych itd. Drugą grupę stanowią **specjaliści w innych zawodach**, na przykład nauczyciele. Dla nich informatyka jest środkiem, narzędziem do rozwiązywania problemów z ich dziedziny. Chcą więc korzystać z niej maksymalnie ekonomicznie, to znaczy najtaniej, w najkrótszym czasie i najlepiej spożytkować możliwości, jakie ona stwarza. Czym innym jest informatyka dla kolejnej grupy **szerokich warstw społeczeństwa:** uczniów, gospodyń domowych, osób korzystających z państwowych środków transportu, usług itd. Spotykają się z nią wszędzie. Na kolei, w biurze rezerwacji miejsc, w bankach – gdzie stanowi instrument codziennego użytku. Grupą, która nie jest rozłączna z poprzednimi, ale którą wyodrębnić należy, są **decydenci.** Francuzi nazywają ich "responsable institutionale". Na nich spoczywa odpowiedzialność moralna, kulturalna, kulturotwórcza – odpowiedzialność za wdrożenie informatyki. I to oni właśnie powinni umieć przewidzieć efekty, jakie informatyka wywrze na życie społeczeństwa.

– Co jest najważniejszym czynnikiem potrzebnym do uzyskania tych efektów?

– Wybitny uczony radziecki, specjalista w dziedzinie informatyki i automatyzacji, akademik Głuszkow sformułował kilka zasad dotyczących wprowadzania informatyki (w przedsiębiorstwie, szkole, resorcie, kraju). Pierwsza z nich – zasada przywódcy – mówi, że żaden zautomatyzowany system, żadna technika informatyczna nie zostaną wprowadzone tam, gdzie osoba odpowiedzialna (przywódca) nie będzie osobiście nimi zainteresowany, nie będzie ich znał i rozumiał. Dlatego na przykład w szkole, bez poparcia i czynnego zaangażowania dyrektora, sam entuzjazm nauczyciela--informatyka nie wystarczy do stworzenia laboratorium informatycznego, do wyjścia poza granice koła zainteresowań, poza granice zabaw informatyką i w informatykę.

– Czego szkoła może oczekiwać od informatyki, jakie daje ona korzyści?

– Informatyka daje szkole trzy bardzo ważne rzeczy. Pierwszą z nich jest **interaktywność,** możliwość prowadzenia dialogu z maszyną, w dodatku w czasie rzeczywistym. Dla dziecka, które się uczy, ma to kapitalne znaczenie. Drugą korzyścią jest ułatwienie **indywidualizacji nauczania.** Uczeń może się oddzielnie, w "swoim" tempie uczyć, mając swoje indywidualne miejsce pracy przy "swojej" klawiaturze. Z kolei nauczyciel sam nie może przekazać tyle dynamiki, ile daje jej **grafika** komputerowa. Grafika ta powinna być bardzo prosta (ale nie uboga!), dostępna i podobna do tej, którą człowiek posługuje się zazwyczaj. Znany jest eksperyment przeprowadzony przez japońskich psychologów, badających dwie grupy uczniów. Jednej z nich przez 5 lat "podawano informatykę", drugiej – nie. Dzieciom z obu grup kazano narysować drzewo. Pierwsze narysowały drzewa tak, jak je widziały na ekranie, drugie – jak widziały w życiu. To powinni wiedzieć autorzy programów.

– Kto powinien tworzyć programy, auczyciele amatorzy czy profesjonaliści?

– Odpowiedź dziś jest jedna. Powinni robić to informatycy i nauczyciele – razem, wspólnie. Jedni bowiem doskonale piszą programy, drudzy znają dydaktykę swojej dyscypliny. Informatyk nigdy nie będzie znał fizyki tak, jak fizyk, fizyk z kolei nigdy nie zna tak informatyki, jak informatyk. Jest to zrozumiałe. Z kolei żaden z nich nie zna tak dobrze psychiki dziecka jak psycholog. Tak więc tworzenie programów jest pracą zespołową, którą muszą wykonywać wspólnie informatycy, nauczyciele i psychologowie. We Francji produkcją programów zajmuje się odpowiedni sektor państwowy. Następnie wszystkie programy kierowane są do specjalnie utworzonego Centrum Dokumentacji Programów Dydaktycznych, gdzie podlegają wstępnej ocenie i skąd rozprowadzane są do szkół. Każda szkoła może bezpłatnie otrzymać dowolny program, napisany w dowolnym języku (Logo, Pascal, Basic itd.). Nie są to programy amatorskie, są dopracowane pod względem programistycznym i dydaktycznym. A jednak budzą one pewne obawy, na przykład generalny inspektor szkół francuskich widzi w nich wyłącznie zamknięty produkt, w którym nie ma miejsca na indywidualne podejście nauczyciela.

– Jakich programów jest najwięcej?

– Trzymając się przykładu francuskiego – najwięcej jest programów matematycznych, do nauki języków obcych oraz francuskiego, a więc z tych przedmiotów, gdzie ważne jest powtarzanie wiadomości i ich utrwalanie.

– Wśród programów można wyodrębnić pewne typy o podobnej organizacji, zbliżonym scenariuszu, choć przeznaczone do wspomagania nauczania różnych przedmiotów. Ma to ścisły związek ze sposobem wykorzystania. Jakie są to sposoby?

– Najbardziej popularną, choć nie najłatwiejszą w realizacji, jest tak zwana metoda nauczycielska (tutorial). Komputer występuje tu w roli nauczyciela, korepetytora. Cały materiał podzielony jest na niewielkie, logicznie uszeregowane porcje. Uczeń odczytuje je z ekranu, po zapoznaniu się z każdą "stroną" odpowiada na pytanie, czy dany fragment materiału zrozumiał. Jeśli nie – otrzymuje dodatkowe informacje, wyjaśnienia, szczegółowe wskazówki. I dopiero po zrozumieniu danej porcji może przejść do następnej. Jak wspomniałem, jest to metoda trudna. Zawsze istnieje wiele dróg uzyskania określonych wiadomości, wiele możliwych odpowiedzi na dane pytanie. Wszystkie te rozgałęzienia bardzo trudno jest uwzględnić w programie. Dlatego rola nauczyciela przy korzystaniu z tak skonstruowanych programów i tej metody nauczania wspomaganego komputerem polega na kontrolowaniu, czy uczeń nie ma kłopotów ze zrozumieniem. I dopiero po ich

stwierdzeniu nauczyciel ingeruje, włącza się. Druga metoda, choć bardziej praktyczna (drill and practice), nie zakłada też zbyt wielkiej aktywności ucznia. Komputer daje mu serię ćwiczeń o różnym stopniu trudności, które należy rozwiązać (bądź tylko wybrać prawidłową odpowiedź spośród kilku możliwych). Po podaniu poprawnej odpowiedzi uczeń przechodzi do następnego, trudniejszego ćwiczenia. Coraz częściej różnica między powyższymi metodami zaciera się, łączy się je w jedną. Uczeń otrzymuje stronę (ekran) tekstu z nowymi wiadomościami, a następnie przykłady sprawdzające zrozumienie tego materiału. Jest więc to w pewnym sensie **komputerowa lekcja:** komputer wykłada pewien materiał i sprawdza stopień opanowania go przez testy, zadania, które uczeń rozwiązuje pod kontrolą nauczyciela.

W naukach przyrodniczych – a więc biologii, fizyce, chemii – najczęściej stosuje się **modelowanie, symulację.** Tworzony jest pewien model danego zjawiska, a więc o jego działaniu decyduje uczeń, który musi dany problem rozwiązać. W zależności od tego, jak go rozwiąże – model pracuje lepiej lub gorzej.

Najciekawsza i mająca ogromne perspektywy metoda polega na **kierowaniu** przez komputer **procesem uczenia się.** Uczeń, student chce się czegoś nauczyć. Podchodzi do komputera, z wykazu zagadnień wybiera to, które go najbardziej interesuje. Wtedy komputer pyta, co wie już na dany temat, sprawdza jego wiadomości wstępne. Załóżmy, że interesują nas równania różniczkowe. Komputer sprawdza, co do tej pory wiemy o nich. Otrzymujemy do wykonania serię testów, komputer ocenia naszą wiedzę i informuje, w jaki sposób należy ją uzupełnić, czego się nauczyć, z czego korzystać, jakie wykonać ćwiczenia. Kieruje więc on naszym uczeniem się. Mówi, czego nie umiemy. Po upływie określonego czasu możemy poddać się kolejnej serii testów, dowiedzieć się, jak opanowaliśmy zagadnienie, czego nauczyliśmy się. Jest to jednak metoda bardzo trudna, wymagająca przygotowania niezwykle dopracowanych programów. Właściwie powstało dotychczas niewiele takich prac, głównie w USA, Japonii i Wielkiej Brytanii.

– Wynika z tego, że program nie powinien pozwolić na bierną postawę ucznia, powinien umożliwić mu podejmowanie decyzji o dalszym jego przebiegu, przede wszystkim – zainteresować.

– Oczywiście! Program musi być ciekawy. Obraz na ekranie nie może zawierać zbyt wiele informacji tekstowej. Tego uczeń nie lubi i nie będzie w stanie opanować, ba – szybko przestanie uważać. Skuteczne są dobre gry dydaktyczne, w których uczeń znajduje elementy zabawy i nauki. Ale powstaje niestety także bardzo wiele programów tak zwanych edukacyjnych, które nie są wartościowe. Na przykład we Francji z 400 programów pozytywnie oceniono tylko 40, w USA z 2000 programów przydatnych okazało się tylko 10%. Oczywiście przydatnych z punktu widzenia dydaktyki. Autorami zdyskwalifikowanych programów byli zawodowi programiści, a nie pedagodzy. Zły program nie przewiduje bezpośredniego kontaktu ucznia i nauczyciela z komputerem. Niedobry jest także program, który odrzuca nauczyciela i ucznia od komputera, kiedy okazuje się, że to samo co robi komputer, może zrobić uczeń przy pomocy ołówka i kartki papieru.

– Mówił Pan o wykorzystaniu komputera w nauczaniu różnych przedmiotów. Pozostała jeszcze cała sfera, którą Jerszow nazwał "drugą alfabetyzacją".

– Tak, "second literacy", "computer literacy", czyli podstawy kultury informatycznej, zwane komputerową alfabetyzacją, należy wprowadzać na rozmaite sposoby. Wszyscy powinni mieć możliwość zapoznania się z elementami i podstawami informatyki, z jej metodami. Na Zachodzie bardzo popularne stało się ostatnio hasło "Computer on every desk" – komputer na każdym stole. Bardzo silny nacisk na "uczenie mikrokomputerów" wywierają dwie grupy. Jedną z nich są producenci sprzętu i jest to oczywiste. Drugą grupę stanowią natomiast rodzice uczniów przekonani, że ich dzieciom – jeśli znać będą mikrokomputery – nie zagrozi w przyszłości bezrobocie. Jest to bardzo ciekawe zjawisko socjologiczne. Stąd też m.in. wynika tak duża liczba komputerów w domach, np. we Francji – powyżej 1 mln.

Zasada przywódcy – mówi, że żaden zautomatyzowany system, żadna technika informatyczna nie zostanie wprowadzona tam, gdzie osoba odpowiedzialna ("przywódca") nie będzie osobiście nią zainteresowana, nie będzie jej znała i rozumiała

Komputerową alfabetyzację wprowadzać można także na kilka sposobów, przy czym są to w zasadzie dwa podejścia: techniczne i pragmatyczne. W pierwszym z nich dzieci otrzymują wiedzę w dziedzinie algorytmizacji, programowania, budowy i obsługi maszyny. Wpaja im się techniki obliczeniowe. Metoda ta ma wielu przeciwników, którzy mówią: „nie uczymy obsługi i działania radia, telewizora, telefonu, samochodu. Dlaczego mamy robić to z komputerem?" Nie jest to jednak tak jednoznaczne. Podejście pragmatyczne za punkt wyjścia bierze następujący fakt. Dziś uczymy dzieci, które swoją pracę zawodową rozpoczną w roku 2000. To, czego nauczą się one o współczesnych mikrokomputerach, współczesnych językach programowania, w przyszłości będzie nieprzydatne, przestarzałe, wręcz śmieszne. Należy więc uczyć czegoś innego, nie budowy mikrokomputerów, nie języków programowania, lecz systemów obsługi danych, edycji tekstów, zastosowania komputera w określonych dziedzinach.

Ciekawe jest, że o wiele wcześniej traktowano informatykę raczej jako środek w nauczaniu innych dyscyplin niż samodzielny przedmiot. W Wielkiej Brytanii w narodowym programie "Komputer i nauczanie", który realizowany był w latach 1973-1977, komputer był tylko środkiem nauczania. Dopiero w "Programie Edukacji Mikroelektronicznej" (1981-85) znalazła się "computer literacy" jako fakultatywny przedmiot nauczania, który mogą wybierać uczniowie 15-letni. Podobnie było we Francji. W programie – czy też eksperymencie 58 liceów (lata 1970-78) – uczono informatyki jako środka w nauczaniu innych dyscyplin, takich jak geografia, matematyka, historia, literatura. W programie "100 000 mikrokomputerów w szkołach" (opracowanym w 1983 r., a realizowanym od 1985 r.) uczniowie 15-18-letni mają możliwość wybrać informatykę jako przedmiot nauczania.

– Czego możemy spodziewać się w przyszłości?

– No cóż, możliwe są trzy scenariusze. Pierwszy – pesymistyczny: w wieku dwudziestym pierwszym nie będzie szkół i nauczycieli, bowiem wszystkie dzieci będą się uczyć w swoich domach. Stworzone zostaną olbrzymie bazy danych ze wszystkich przedmiotów, każde dziecko będzie oceniane u siebie w domu, przez swój komputer. Każde dziecko stanie się egoistą, osobą aspołeczną, zamknie się w sobie. Jego całe życie przeminie w domu przed telewizorem. Wszyscy ludzie ulegną wpływowi informatyki. Człowiekowi nie pozostanie nic własnego, osobistego, stanie się częścią dużego systemu.

Scenariusz optymistyczny: w następnym stuleciu szkoła oczywiście przetrwa, lecz zmienią się jej funkcje. Wprawdzie dzieci uczyć się będą przede wszystkim w domu, w sposób zindywidualizowany, ale szkoła stanie się miejscem kontaktów międzyludzkich, spotkań towarzyskich. Wprowadzi się takie przedmioty, jak filozofia czy retoryka, wymiana poglądów. Natomiast komputer dysponując ogromnymi bankami danych, będzie w procesie nauczania pomagał.

Trzecie podejście ma charakter liberalny. Gdy nauczymy się wykorzystywać wszystkie możliwości komputerów, nauczymy się również nimi kierować, kierować procesem nauczania z pomocą komputerów. Wówczas zorientujemy się, jak niezwykle użyteczne mogą być komputery. Już na przykład w tym roku dzieci marokańskie mogły korzystać – dzięki łączności satelitarnej – z centralnych baz danych szkół i instytutów francuskich. Rzecz polega więc na dobrym zarządzaniu i oswojeniu informatyki.

JAKOŚĆ A NIE ILOŚĆ

Z doc. dr hab. JANEM MADEYEM, dyrektorem Instytutu Informatyki UW, o pracy w Wang Institute rozmawia Władysław Majewski.

Co to jest Wang Institute, uczelnia czy ośrodek badawczy?

Przede wszystkim uczelnia, choć trudno placówkę tak nietypową nawet w USA oceniać w krajowych kategoriach.

Warto kilka zdań poświęcić jej twórcy i fundatorowi.

Dr An Wang przyjechał do USA w 1944 r. z Shanghaju i uzyskał w Massachusetts Institute of Technology (MIT) doktorat z fizyki. Wynalazł wówczas tzw. rdzenie ferrytowe, z których przez prawie 20 lat budowano pamięci operacyjne komputerów.

W 1951 r. założył własną firmę komputerową specjalizującą się w systemach do przetwarzania tekstów, która na wiele lat zdominowała rynek. Również w Polsce pracuje wiele komputerów tej firmy, głównie starszej generacji. Później asortyment produkcji rozszerzył się. Kibice na pewno znają komputery Wanga z wielu transmisji sportowych, podczas których podają one wyniki. Ostatnio zaczęto produkować tzw. Wang Professional Computer, łączący znakomite przetwarzanie tekstów z zaletami dobrego mikrokomputera przewyższającego IBM PC.

Obecnie firma zatrudnia ok 30 000 pracowników. Jej właściciel, będąc jednym z najbogatszych ludzi w USA, pracuje jednak codziennie po kilkanaście godzin, uczestniczy w nowych projektach technicznych, znajduje czas, by zapoznać się z tematami prac studentów.

Wang Institute powstał w 1979 r. i jest jedną z dwóch uczelni amerykańskich specjalizujących się w inżynierii oprogramowania. School of Information Processing to jej pierwszy wydział. Placówka jest stale dotowana przez fundatora, który chciałby przejść do historii nie tylko jako przedsiębiorca.

Dewizą szkoły jest "jakość nie ilość". Przyjmowani są studenci z pierwszym stopniem naukowym z informatyki zdobytym w innych uczelniach oraz co najmniej rocznym udokumentowanym stażem pracy. Muszą zdać trudny egzamin wstępny m.in. z matematyki i przedstawić samodzielnie wykonany projekt techniczny. Oceniany jest także opis tego projektu. Niekoniecznie ma być to publikacja, ale musi dowodzić umiejętności posługiwania się językiem naukowym. Studentami rocznych studiów w Instytucie zostaje jedynie ok. 50 ubiegających się. Studentów i wykładowców czeka rok ciężkiej harówki. Zajęcia są bardzo zróżnicowane: od wykładów o charakterze podstawowym – jak np. metody formalne inżynierii oprogramowania, metodologia programowania, architektura komputerów, do praktycznych aspektów sterowania dużymi projektami – metody zarządzania, metody pracy zespołowej przy projektach. Ponadto zespoły studentów samodzielnie opracowują bardzo złożone projekty.

Atmosfera jest znakomita.

Studenci pracują bardzo ciężko dnie i noce, z sobotami i niedzielami włącznie. Wszyscy podkreślają, że studia te wymagają całkowitego oddania się pracy. Równocześnie stworzono znakomite warunki do tak intensywnego wysiłku. Wyposażono Instytut w sprzęt rzadko spotykany nawet w czołowych uniwersytetach amerykańskich.

Jest kilka dużych komputerów oraz naprawdę wiele sprzętu mikro, głównie typu Apple II, MacIntosh, Lisa, IBM PC i oczywiście Wang PC.

Te ostatnie mają po 640 KB RAM i 30 MB na twardym dysku.

Są to typowe komputery osobiste, choć cechami użytkowymi przewyższają komputery, jakimi dysponuje nasz Instytut w Warszawie.

Każdy pracownik Wang Institute ma w swoim pokoju od 1 do 3 takich urządzeń. Mogą one pracować jako końcówki jednego z dużych systemów komputerowych Instytutu lub sieci krajowej. Wygląda to tak, że po włączeniu systemu pada pytanie: Czym mam być? Wang PC z wirtualnym dyskiem 100 MB, IBM PC, czy może końcówką do komputera VAX sterowanego systemem operacyjnym np. UNIX?

Jest to wielka wygoda, ponieważ używa się tej samej klawiatury i tego samego ekranu, choć trzeba przyznać, że dopóki się nie dojdzie do wprawy, można mieć kłopoty. W różnych systemach niekiedy ten sam klawisz ma bardzo różne znaczenie...

Takim właśnie komputerowym okienkiem w świat dysponuje każdy student. Zmienia to zupełnie styl współpracy:

wykładowców i studentów, nie ma żadnych tablic ogłoszeń, zastępuje je bardzo rozbudowany system elektronicznej poczty. Po przyjściu do pracy włącza się komputer, na którego ekranie pojawia się strukturalnie zorganizowany wykaz wiadomości, pozostawionych dla danej osoby.

Komunikaty można wysyłać do pojedynczych osób lub do różnych grup, zachowując w razie potrzeby poufność korespondencji.

Równocześnie Instytut jest podłączony do światowej sieci akademickiej, obejmującej USA, Kanadę i Europę Zachodnią. Można wysyłać wiadomość na kraniec świata i po kilku godzinach otrzymać odpowiedź, która przechodzi przez kilkanaście komputerów. Sieć umożliwia takie upowszechnienie publikacji, a więc przekazy wiedzy. Ta forma obiegu informacji zaczyna zastępować czasopisma naukowe. Abonenci sieci mogą prenumerować np. komunikaty o systemach operacyjnych. Otrzymają wszystkie ukazujące się z tej dziedziny informacje. Komunikaty wprowadzane do sieci mogą być dowolnie długie, często są to np. teksty programów lub liczące setki stron zestawy danych np. surowych wyników pomiarów.

Wadą takiej sieci jest brak filtrów a właściwie samokontroli środowiska naukowego nad informacjami wprowadzanymi do obiegu. Siecią można upowszechniać wszelkie głupstwa, to się zdarza powodując powszechną irytację. Często ten sam artykuł wysyłany bywa w ramach różnych grup tematycznych. Oto inny przykład funkcjonowania sieci.

Rok temu społeczeństwo amerykańskie było zbulwersowane sprawą zmiany receptury coca-coli. Zdaniem socjologów właśnie sieć komputerowa umożliwiła szybkie rozpowszechnienie apelu o bojkot nowego napoju

i wymuszenie powrotu do tradycyjnego smaku.

Rozumiem, że sieć stała się narzędziem agitacji wrogiej firmie Coca-Cola Co, i że w ten sposób nadużyto wolności posługiwania się nią?

Właśnie tak. We wszystkich grupach tematycznych sieci pojawiły się dyskusje o coca-coli.

A czy nie wzbudziło to niepokoju władz uniwersytetów, finansujących tę "zabawę"?

Stale pojawiają się żądania uporządkowania sprawy, ograniczenia tematyki, ale nic nie wskazuje na to, by zdecydowano się na ten krok.

Czy sieć daje takie możliwości dostępu np. do wielkich baz danych?

Nie każda, ale niezależnie od tego, są inne, dla nas szokujące wręcz możliwości. Dla przykładu, wykładając w tym roku systemy operacyjne musiałem korzystać z pewnego oprogramowania, które kilka lat wcześniej zostało pod moim kierunkiem uruchomione przez mojego studenta z Uniwersytetu Kalifornijskiego. Udało mi się go odnaleźć i namówić do wprowadzenia zmian. Następnie

podałem mu swoje tajne hasło uprawniające do pracy na komputerze oraz numer telefonu. To wystarczyło, by mógł siedząc u siebie w pokoju w Dolinie Krzemowej w Kalifornii, na drugim końcu kontynentu, pracować w rzeczywistości na sprzęcie Wang Institute.

Czy mimo takiej pomocy możliwe jest nauczanie systemów operacyjnych po wielu miesiącach pobytu poza USA, zwłaszcza jeśli studentami są osoby z praktyką i często z doktoratami z dziedzin pokrewnych?

Pewna świeżość spojrzenia też się przydaje, ułatwia dostrzeżenie ogólniejszych prawidłowości, przede wszystkim jednak niezbędna jest ciężka praca. Jeśli w Instytucie prowadzę 5 godzin wykładu tygodniowo, to do każdej przygotowuję się przez kilkanaście godzin, traktując dydaktykę jako główny cel mojego tam pobytu. Oczywiście często uczę studentów czegoś, czego poprzedniego dnia sam się uczyłem.

Wciąż jeszcze udaje mi się wywoływać u studentów wrażenie, że znam te systemy lepiej od nich.

Nie ma jednak mowy o powtarzaniu co rok tego samego wykładu: zmiany w tej dziedzinie są zbyt szybkie.

A jak radzą sobie absolwenci Instytutu?

Studia w Wang Institute mają bardzo dobrą reputację, nie jest zresztą łatwo je ukończyć, nie wszyscy dochodzą do uroczystej promocji. I chociaż Instytut mieści się na granicy Massachusetts i New Hampshire w wykupionym przez dra Wanga klasztorze położonym w lesie, nad rzeką, w pełnej izolacji od cywilizacji, jednak miejsce to stwarza dobrą atmosferę do wytężonej pracy.

Biblioteka informatyczna jest oczywiście młodsza od budynku, zawiera jednak wszystkie ważne światowe czasopisma, a każdy profesor ma prawo żądać sprowadzenia dowolnej publikacji.

W okresie letnim odbywają się, poza normalnymi zajęciami, tygodniowe kursy czyli tzw. letni instytut informatyki. Udział w nich kosztuje ok. 1000 dolarów. Jednak pracownicy i studenci mogą w kursie uczestniczyć bez dodatkowych opłat.

Ile kosztują studia roczne?

Ok. 12 000 dolarów. Trzeba też zrezygnować z pracy.

Dla studenta nie mającego sponsora oznacza to poświęcenie ok. 40 000 dolarów. Jeśli niektórzy decydują się na taką stratę, to muszą wierzyć, że dzięki studiom w Wang Institute potrafią ją później szybko odrobić. W USA od nadawanego przez uczelnię stopnia ważniejsza jest pozycja jej i jej absolwentów, a absolwenci Wanga świadczą o uczelni jak najlepiej. Instytut szybko zdobył reputację i absolwenci nie mają kłopotu ze zdobyciem dobrej pracy. Studenci zwykle rekrutują się z największych firm, które dają im urlop i często nawet opłacają ich studia.

Ci, którzy nie mają oparcia w żadnej firmie, mogą podjąć pracę w Instytucie jako asystenci, mają wtedy obowiązek przez 20 godzin tygodniowo pracować na rzecz jednego z profesorów. Jest to zresztą jedyna forma pomocy, jaką profesor tam otrzymuje. Nie ma stanowisk asystentów czy młodszych pracowników naukowych.

Jak zostaje się profesorem?

Trzeba mieć doktorat i duże doświadczenie w inżynierii oprogramowania. Uczelnia nie ma charakteru czysto akademickiego. Wymagana jest praktyka. Bardzo duże znaczenie przywiązuje się do jakości wykładania. Nie ma obowiązku publikowania, choć strona naukowa zaczyna odgrywać coraz większą rolę, zresztą przy tym poziomie słuchaczy bez pracy własnej nie ma mowy o wykładaniu.

Przy przyjmowaniu do pracy i przedłużaniu kontraktu pod uwagę brana jest opinia studentów. Każdy kandydat na profesora najpierw odbywa wykład i spotkanie ze studentami. Rzuca się go na pastwę studentów, którzy go bardzo dokładnie z lewa i prawa oglądają, zadając różne trudne pytania, często niedyskretne. Cały życiorys jest przeglądany i dyskutowany. Po każdym trymestrze (1/3 roku akademickiego) również studenci oceniają każdego wykładowcę. Ocena składa się z dwóch części: odpowiedzi na ok. 30 pytań dotyczących m.in. tematyki, sposobu prowadzenia zajęć, otwartości wykładowcy na dyskusję, jego dostępności. Np. ostatnie trzy pytania brzmią: Czy chciałbyś wysłuchać jeszcze jakiegoś wykładu tego profesora? Czy poleciłbyś go komuś? Czy poleciłbyś ten wykład? Natomiast druga ankieta zawiera komentarze słowne. Są one anonimowe, a wykładowca dostaje syntetyczne opracowanie wniosków.

Ankiety mają znaczenie dla każdego pracownika. Nie we wszystkich bowiem uczelniach amerykańskich istnieją tzw. tenury – dożywotnie zatrudnienie. Najdłuższe są pięcioletnie kontrakty.

Niezależnie od wykładów kursowych odbywają się tzw distinguish lectures – wystąpienia specjalistów z całego świata zapraszanych dla wygłoszenia jednego wykładu. Większość najbardziej znanych postaci światowej informatyki miała już tam takie prezentacje, co pozwala studentom poznać tych ludzi osobiście, wymienić z nimi poglądy itp.

Ilu jest profesorów w Wang Institute?

Na co najmniej rocznych kontraktach odnawianych w ciągu 10 czy nawet 15 lat pracuje zawsze kilku profesorów o znanych nazwiskach. Od ponad 5 lat pracuje tam Bill McKeeman, świetny specjalista od języków programowania i ich kompilatorów. Przez kilka lat pracowała tam Susan Gerhardt, specjalizująca się w systemach takiego wspomagania programowania, by można było wykazywać jego poprawność. Jest tam teraz Phil Barnstein, jeden z najwybitniejszych specjalistów od baz danych.

Ośrodek obliczeniowy jest, jak już wspomniałem, bogato wyposażony, lecz ma bardzo nieliczną kadrę. Składa się ona z dyrektora, osoby odpowiedzialnej za kontakt z użytkownikami, jednego dobrego głównego programisty i kilku operatorów zatrudnionych w niepełnym wymiarze godzin, spełniających też funkcję techników.

Administracja liczy "aż" ok. dziesięć osób, gdyż akcja rekrutacyjna wymaga wiele pracy. Sekretariat jest oczywiście w pełni skomputeryzowany. Maszyn do pisania właściwie się nie spotyka.

Jak Pan tam trafił?

Pracowałem przez półtora roku w Uniwersytecie Kalifornijskim w Santa Cruz. Z inicjatywy McKeemana, który w tym okresie przeniósł się już z Kalifornii do Massachusetts, zaproszono mnie do wygłoszenia referatu. Przedstawiłem wówczas stosowaną przeze mnie metodykę nauczania systemów operacyjnych, co wzbudziło zainteresowanie i spowodowało zaproszenie mnie na letni trymestr. Odtąd wykładałem tam już trzykrotnie.

A jak oceniają Pana studenci?

Z roku na rok lepiej, co utrudnia mi pracę, bo mój wykład jest wybierany przez coraz liczniejszą i lepszą grupę, a ja staram się być dla swoich studentów stale dostępny i dostosować prowadzone zajęcia do ich potrzeb.

Dla wielu studentów jestem ciągle jeszcze pewnym zaskoczeniem, jako przedstawiciel kraju socjalistycznego. Ci, którzy tak odczuwają, są przekonani, że Stany Zjednoczone mają monopol na wiedzę o komputerach.

Czy wyczuwa się tam tak częstą na uczelniach amerykańskich atmosferę rywalizacji?

Raczej nie, jest wspaniała atmosfera wspólnej pracy i zaufania, zakłada się wzajemną uczciwość.

Każdy ma dostęp do wszystkiego i skopiowanie oraz wyniesienie oprogramowania jest stosunkowo łatwe. Nie robi się jednak tego. Wszyscy przestrzegają zasad etyki zawodowej.

Równocześnie nie korzysta się z możliwości utrudniania czy uniemożliwiania dostępu do własnych plików czyli zbiorów danych, a nawet często celowo czyni się je dostępnymi dla innych. Np. każdy ma swój kalendarz na komputerze. I jeśli chcę się z kimś spotkać, to po prostu zaglądam do jego kalendarza i sprawdzam, kiedy jest wolny i kiedy będzie na miejscu.

Zrozumiałe jest, że zaglądanie w cudze pliki, to tak jakby wejście do czyjegoś pokoju i zaglądać w szufladę. Tego się nie robi, choć wszystkie drzwi, tak jak pliki, są zawsze otwarte. Nie występują kłopoty znane z innych dużych uniwersytetów. W Instytucie nie zdarza się, by ktoś tam złamał szyfr, dostał się do cudzych zasobów i namieszał, pozmieniał itp.

Ta atmosfera i zasoby pozwalają w Wang Institute, obok prowadzenia wykładu, czynnie realizować projekty. Tego wciąż nie udaje mi się robić w Warszawie, głównie z powodu braku sprzętu i motywacji u studentów. Tam zmuszam ich do bardzo dużego wysiłku, ale i pomagam im w pracy.

Jaki jest współczesny stan inżynierii oprogramowania?

Ciągle daje się odczuć dystans między teorią a praktyką. Absolwenci Wang Institute niekiedy napotykają trudności, gdy próbują upowszechniać bardziej nowoczesne metody. Opór jest często samoobroną osób, które nie są dostatecznie wykształcone i czują, że stosowanie lepszej metodyki oznacza koniec ich kariery. Znają one jedynie COBOL FORTRAN i nie są w stanie nic innego robić.

Staje się też oczywiste, że metody testowania programów nie są wystarczające.

W USA głośna była sprawa rezygnacji jednego z profesorów-informatyków z udziału w programie wojen gwiezdnych. Motywował on swoją decyzję tym, że nie można oczekiwać, by programy były niezawodne. Jest więc ciągle duża różnica między inżynierią oprogramowania a innymi inżynieriami, wciąż nie ma bezpiecznej metodyki dowodzenia poprawności programów.

Jest to rzecz o tyle trudna, że można wykazać, iż program realizuje to, co jest w inny formalny sposób opisane. Problem polega jednak właśnie na ścisłym określeniu tego, co program ma robić.

Spośród języków programowania, coraz bardziej popularny staje się język C. A to ze względu na sukcesy systemu operacyjnego UNIX i udane połączenie cech języka niskiego poziomu, pozwalającego głęboko wchodzić w strukturę komputera, a także stosować różne triki z możliwością formułowania problemów w nowoczesny, strukturalny sposób. Zainteresowanie budzi prolog, głównie pod wpływem japońskiego programu 5 generacji. ADA nie jest lubiana, ale wszyscy zdają sobie sprawę z jej wagi. W nauczaniu wciąż dominuje PASCAL. FORTRAN i COBOL są w zaniku. Do ustąpienia języka BASIC dojdzie dopiero wtedy, gdy nawet najtańszy mikrokomputer wyposażony będzie w dużo lepsze języki, takie jak LOGO I PASCAL.

Mimo zapracowania, wziął Pan udział w światowym kongresie na temat roli i udziału komputerów w procesie edukacji, zorganizowanym w Norfolk. Wystąpił Pan tam jako przewodniczący zespołu doradczego ministra oświaty ds. stosowania komputerów w nauczaniu?

Byłem tam jedynym Polakiem, ale koszty mego pobytu pokrył dr Wang, a nie minister oświaty. Kongresy takie odbywają się raz na pięć lat w różnych krajach. Uczestniczyło w nich ponad 3000 osób, odbyła się wspaniała wystawa, której nikt z Polski nie oglądał. Na kongresie przyjęto polski referat, lecz go nie wygłoszono. To bardzo smutne, gdyż wystawa grupująca 120 wystawców dawała szansę zapoznania się ze światową sytuacją w zakresie sprzętu, oprogramowania, czasopism, podejścia do spraw nauczania.

SPRAWDZIAN

Dobrze wiemy, jak bardzo nużące jest odpytywanie dzieci z dzielenia, odejmowania, dodawania bądź tak pożądanej znajomości "na pamięć" tabliczki mnożenia. Program "sprawdzian" nie tylko przeegzaminuje dziecko, wystawi ocenę z każdego z typów działań arytmetycznych oraz ocenę łączną, ale i podając przy błędnych odpowiedziach prawidłowy wynik, być może i czegoś nauczy. Jest też i nagroda dla piątkowiczów – jej pełny urok można poznać jednak jedynie dysponując kolorowym monitorem, chociaż wyrazy uznania ze strony komputera nie ominą oczywiście i posiadaczy telewizorów monochromatycznych.

Na samym początku należy określić, ile zadań każdego z rodzajów działań postawi przed nami komputer, im jest ich więcej, tym mniejszy wpływ na wynik mają rozwiązania błędne. Po raz pierwszy proponuję określić ilość zadań na 5.

Do nas należy także decyzja wyznaczenia przed każdym pierwszym z danego rodzaju działań, stopnia trudności. Trudność "1" oznacza działanie komputera na liczbach jednocyfrowych, "2" – na jedno- i dwucyfrowych itd. Pozwala to na odpowiednie dostosowanie trudności zadań do umiejętności odpowiadającego. Np. w I klasie szkolnej wystarczy raczej umiejętność działań na liczbach jednocyfrowych, już w trzeciej – dodawanie i odejmowanie powinno być wykonywane na liczbach przynajmniej dwucyfrowych (a pod koniec roku i inne rodzaje działań).

Wszystkie zadania z poszczególnych działań wyznaczane są losowo i używają jedynie liczb całkowitych. Losowość doboru składników działań może niekiedy powodować ich powtarzanie. Również losowo odbierana jest kolejność poszczególnych rodzajów zadań.

Na samym początku programu zdefiniowane zostały używane zmienne, a niektórym z nich przypisano powtarzające się w programie słowa. Tego rodzaju wyodrębnienie słownika ułatwia modyfikację programu i jest elementem przejrzystego stylu programowania, podobnie jak staranny komentarz.

Program działa również na ZX 81 oraz jego amerykańskich wersjach: TIMEX 1000 i 1500. Ich posiadacze powinni pominąć jedynie polecenia związane z kolorem (INK, PAPER), dźwiękiem (BEEP), oraz zapisać wszystkie polecenia w oddzielnych liniach. Linie zawierające całe grupy poleceń po instrukcji warunkowej IF można samodzielnie zamienić na podprogramy. (Np. 430 IF i (r) = 0 THEN GOSUB 800 a dalej PRINT itd, 810 INPUT j (r) i 820 RETURN). Program będzie działał nieco wolniej niż na Spectrum, konieczna jest jednak pamięć większa niż 2K w podstawowej wersji ZX 81 lub TIMEX-a.

MAREK

```
   5 REM >>>Sprawdzian
            z arytmetyki<<<
  10 DIM i(4): REM ile prob?
  15 DIM j(4): REM jak trudne?
  20 DIM t(4): REM punkty
  25 DIM d$(4,11): REM dzialania
  30 LET d$(1)="DODAWANIE  "
  35 LET d$(2)="ODEJMOWANIE"
  40 LET d$(3)="MNOZENIE   "
  45 LET d$(4)="DZIELENIE  "
  50 LET w$="Poprawny wynik:"
  55 LET t$="TAK"
  60 LET n$="NIE"
  65 LET s$="*  Sprawdzian  *"
  70 LET u$="Nie zartuj !"
  75 LET r$="OCENA"
  80 LET z$="+-*/"
  85 CLS : INK 0: PAPER 7
  90 REM >>>czolowka<<<
 100 FOR x=6 TO 1 STEP -1
 110  FOR z=21 TO 0 STEP -1
 115   BEEP .05,z*(x/2)
 120   PRINT INK x;AT z,8;s$
 130  NEXT z
 140  FOR z=21 TO 1 STEP -1
 150   PRINT AT z,8;
      "                "
 160 NEXT z: NEXT x
 200 REM >>>wstep, ile zadan?
 210 PRINT INK 4;AT 3,23;",";
AT 4,5;"Po ile chcesz zadan ?";
AT 6,8;"Najmniej po 3 !"
 220 GO SUB 700: REM melodyjka
 230 INPUT a:
     IF a<3 THEN GO TO 310
 240 PRINT INK 4;AT 3,23;" ";
       AT 0, 0;"*";a;"*";
       AT 0,29;"*";a;"*"
 250 FOR q=1 TO a:
     BEEP .2,24: NEXT q
 260 GO SUB 1000: REM scieranie

 270 REM >>>kratki na ekranie
 280 FOR x=9 TO 21
 290  BEEP .01,x
 300  PRINT INK 3;   AT x, 0;"█"
;      AT x,15;"█";AT x,31;"█"
 310 NEXT x
 320 FOR x=0 TO 31
 330  BEEP .01,x+21
 340  PRINT INK 3;   AT 21,x;"█"
;      AT 15,x;"█";AT  9,x;"█"
 350 NEXT x
 360 PRINT INK 4;
       AT 17,3;t$;AT 17,18;t$;
       AT 11,3;t$;AT 11,18;t$
 370 PRINT INK 2;
       AT 17,10;n$;AT 17,26;n$;
       AT 11,10;n$;AT 11,26;n$
 380 PRINT INK 0;
    AT 16,4;d$(3);AT 16,19;d$(4)
;   AT 10,3;d$(1);AT 10,18;d$(2)
 390 REM >>>rozwiazywanie zadan
 399 REM >>>Jakie dzialanie?
 400 LET r=INT (RND*4)+1
 410 IF i(1)+i(2)+i(3)+i(4)=4*a
     THEN GO TO 5000
 420 IF i(r)=a THEN GO TO 400
 429 REM >>>Jak trudne zadania?
 430 IF i(r)=0 THEN PRINT
      AT 2,11;d$(r);
INK 2;AT 4,7;"   JAK TRUDNO?   "
     ;AT 5,7;"   1 , 2 ALBO 3  "
     ;AT 6,7;"                 "
     : GO SUB 700:
       INPUT j(r): BEEP .2,24
 435 GO SUB 1000: REM scieranie

 439 REM >>>losowanie liczb
 440 LET i(r)=i(r)+1
 450 LET b=INT (RND*10↑j(r))+1
 460 LET c=INT (RND*10↑j(r))+1
 470 IF b= 1 OR c=1 OR b=10 OR
        c=10 OR (r=2 AND b<c)
     THEN GO TO 450
 480 PRINT AT 2,11;d$(r)
 485 IF r=4 THEN GO TO 600
 490 PRINT AT 4,10;b;" ";z$(r);
     " ";c;" = ";
 500 LET wynik=VAL (STR$ (b)+
               z$(r)+STR$ (c))
 505 LET e$=STR$ (wynik)
 510 BEEP .2,12: BEEP .2,12:
     BEEP .2,16: BEEP .2,16
 520 INPUT f$
 530 PRINT f$
 540 IF LEN f$>LEN e$ THEN
      PRINT AT 6,10;u$:
      FOR q=1 TO 15: BEEP .5,8:
      NEXT q: PAUSE 100:
      GO TO 400: REM nie zartuj!
 550 IF f$=e$ THEN
      LET t(r)=t(r)+1:
      PRINT AT 13+6*(r>2),
      4+15*(r/2=INT (r/2));t(r);
      AT  6,14;t$:
      BEEP .5,12: PAUSE 100:
      GO TO 400: REM dobrze
 560 PRINT AT 5,14;n$;
           AT 6, 7;w$;e$;
           AT 13+6*(r>2),
           11+16*(r/2=INT (r/2))
           ;i(r)-t(r): REM zle!
 570 BEEP .2,23: BEEP .2,23
 580 PAUSE 200: GO TO 400
 600 LET e=b*c: REM dzielenie
 610 PRINT AT 4,10;e;" : ";b;
     " = ";
 620 LET e$=STR$ c
 630 GO TO 510
 700 BEEP .1,0: BEEP .1, 4:
     BEEP .1,7: BEEP .3,12:
     RETURN : REM melodyjka
 999 REM scieranie gory ekranu
1000 FOR x=5 TO 28
1010 PRINT AT 2,x;" ";AT 4,x;" "
          ;AT 5,x;" ";AT 6,x;" "
1020 NEXT x: RETURN
5000 CLS : REM  cenzurka
5010 PRINT INK 1;
        AT 1, 8;s$;AT 3,11;"*";
        AT 3,13;r$;AT 3,19;"*"
5020 FOR x=0 TO 31
5025  BEEP .1,x
5030  PRINT INK 5;  AT  6,x;"*";
        AT 16,x;"*";AT 20,x;"*"
5040 NEXT x
5050 PRINT INK 0;  AT  8,2;d$(1)
    ;AT 10,0;d$(2);AT 12,3;d$(3)
    ;AT 14,2;d$(4);AT 18,5;r$
    ;" OGOLNA:"   ;AT 17,13;","
5055 GO SUB 700: REM melodyjka
5060 FOR x=8 TO 14 STEP 2
5070  PRINT AT x,12;r$;":"
5075  GO SUB 700: REM melodyjka
5080 NEXT x
5099 REM >>>oceny
5100 LET x= 8: LET y=18
5130 LET i=t(1): GO SUB 5400
5150 LET x=10
5170 LET i=t(2): GO SUB 5400
5190 LET x=12
5210 LET i=t(3): GO SUB 5400
5230 LET x=14
5250 LET i=t(4): GO SUB 5400
5270 LET x=18
5280 LET i=INT ((t(1)+t(2)+t(3)+
       t(4))/4): GO SUB 5400

5300 PAUSE 300
5310 IF h>=80 THEN GO TO 5600:
     REM >>>laurka
5320 GO TO 5500: REM  koniec
5399 REM >>>ocena
5400 LET h=(i*100)/a
5410 IF h>=80 THEN PRINT INK 4;
       AT x,y;"Bardzo dobry."
5420 IF h< 80 THEN PRINT INK 1;
       AT x,y;"        Dobry."
5430 IF h< 60 THEN PRINT INK 3;
       AT x,y;" Dostateczny."
5440 IF h< 40 THEN PRINT INK 2;
       AT x,y;"Niedostateczny"
5445 GO SUB 700: REM melodyjka
5450 RETURN
5510 PRINT INK 6;AT 1,4;
      "* Sprawdzian skonczony *"
     ;AT 0,21;",": REM koniec
5515 FOR q=1 TO 30: BEEP .1,8:
     NEXT q: STOP
5600 CLS : PRINT AT 9,8;
    "Jak Ci na imie ?";
        AT 10,21;"'": REM laurka
5610 GO SUB 700: REM melodyjka
5620 INPUT i$
5625 BEEP .3,12: BEEP .1,7:
     BEEP .1, 4: BEEP .1,0
5630 CLS
5640 FOR v=1 TO 3
5650  GO SUB 5700: REM Brawo!
5660 NEXT v
5670 GO TO 5510: REM koniec
5700 PRINT AT 0,0
5705 LET k=INT (RND*7)
5707 LET i=INT (RND*7)
5715 IF k=i THEN GO TO 5705
5720 FOR x=150 TO 373
5723  BEEP .01,INT (x/10)
5730  PRINT INK k;"*";
5740  PRINT INK i;i$(1);
5750  PRINT INK k;"*";
5760 NEXT x

5763 LET l=INT (RND*7)
5765 IF l=k OR l=i THEN
      GO TO 5763
5770 LET m$="  * BRAWO "+i$+" *
"
5780 LET y=(32-LEN m$)/2
5790 PRINT INK l;AT 9,y;m$
5800 FOR z=y TO (y+LEN m$)-1
5810  PRINT INK l;AT 10,z;" ";
                  AT  8,z;" "
5820 NEXT z
5825 BEEP .2,14: BEEP .2,14:
     BEEP .2,14: BEEP .2,19:
     BEEP .2,19: BEEP .2,19
5830 PAUSE 100: RETURN
```

```
*5*      *  Sprawdzian  *      *5*

           MNOZENIE

        77 * 52 = VAL$ 77*52

          Nie zartuj !
```

LOAD "ORTOGRAFIA"

Kiedy w 1982 r. ukazał się Raport Klubu Rzymskiego "Mikroelektronika a społeczeństwo", jeden z publicystów zaproponował: "Nie stać nas na pościg sprzętowy, pokażemy więc światu, że mamy coś do powiedzenia w oprogramowaniu. Może tą drogą uda się wymienić rodzimą myśl na zachodni sprzęt". Poddałem wtedy w wątpliwość możliwości polskich informatyków w tym względzie, za co zostałem ostro skarcony.

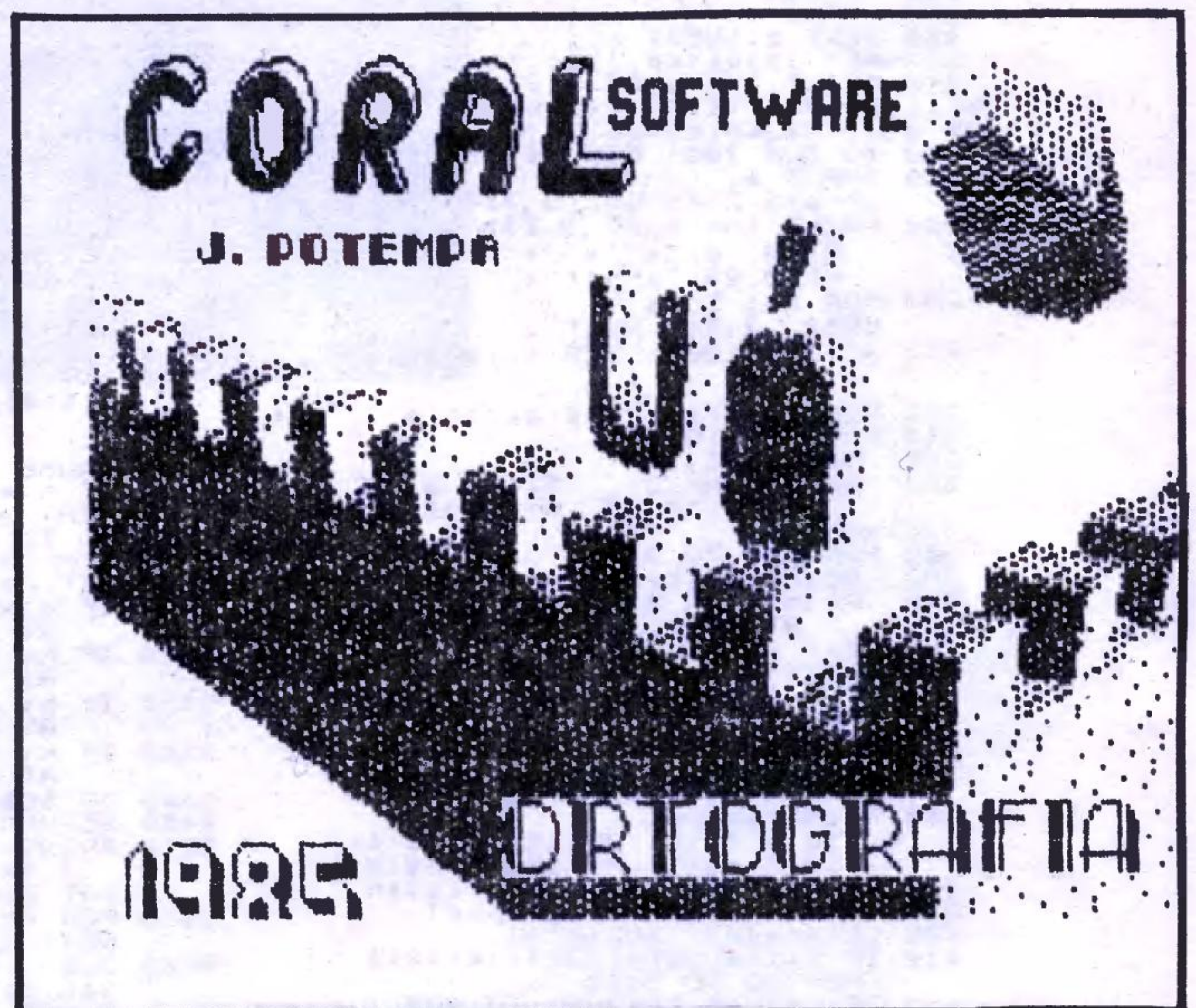
Dzisiaj po prawie 4 latach nadal podtrzymuję swoje wątpliwości. Tylko nieliczne ojczyste programy trafiają na masowy rynek, ot najwyżej kilka z nich zasługuje na omówienie. Należy do nich ORTOGRAFIA Jacka Potempy.

Zadaniem uczącego się za pomocą tego programu jest ocena ortograficznej poprawności pojawiających się na monitorze wyrazów i cała "zabawa" sprowadza się w gruncie rzeczy do naciskania dwóch klawiszy.

Menu programu "ORTOGRAFIA" obejmuje więc pisownię "u" i "ó", "rz" i "ż", "ch" i "h", "z" i "s", pisownię wyrazów dużą literą, z nie – razem czy osobno, "i" i "j" po spółgłoskach, czy wreszcie pisownię "ą" i "ę", "-om", "-em" i "-oń", "-en". Test obejmować może każdą z wymienionych zasad z osobna lub całość ortografii. Istotnym urozmaiceniem jest możliwość ustalenia limitu czasowego na odpowiedź w przedziale 0-10 sekund.

Główną zaletą "ORTOGRAFII" jest fakt istnienia tego programu. Jego konstrukcja sprzyja jednak utrwalaniu się złych nawyków. Szczególnie u osób obdarzonych dobrą pamięcią wzrokową dłuższe obcowanie z "Ortografią" powoduje utrwalenie się w pamięci złej pisowni. Kiedy bowiem uczeń oceni np. że słowo "brzdonc" napisane jest błędnie – na ekranie nie pojawia się już to samo słowo napisane poprawnie: dostajemy wyłącznie kolejny plus. Całe szczęście, że po błędnej ocenie poprawności pisowni wyrazu pojawia się on po chwili ponownie – już napisany bezbłędnie.

Rozwiązaniem idealnym, choć w naszych warunkach przedwczesnym, byłby syntetyzator mowy. Zadaniem uczącego się byłoby napisanie na ekranie usłyszanego wyrazu. I dopiero w tym momencie zaczynałby się problem, tym razem na serio.

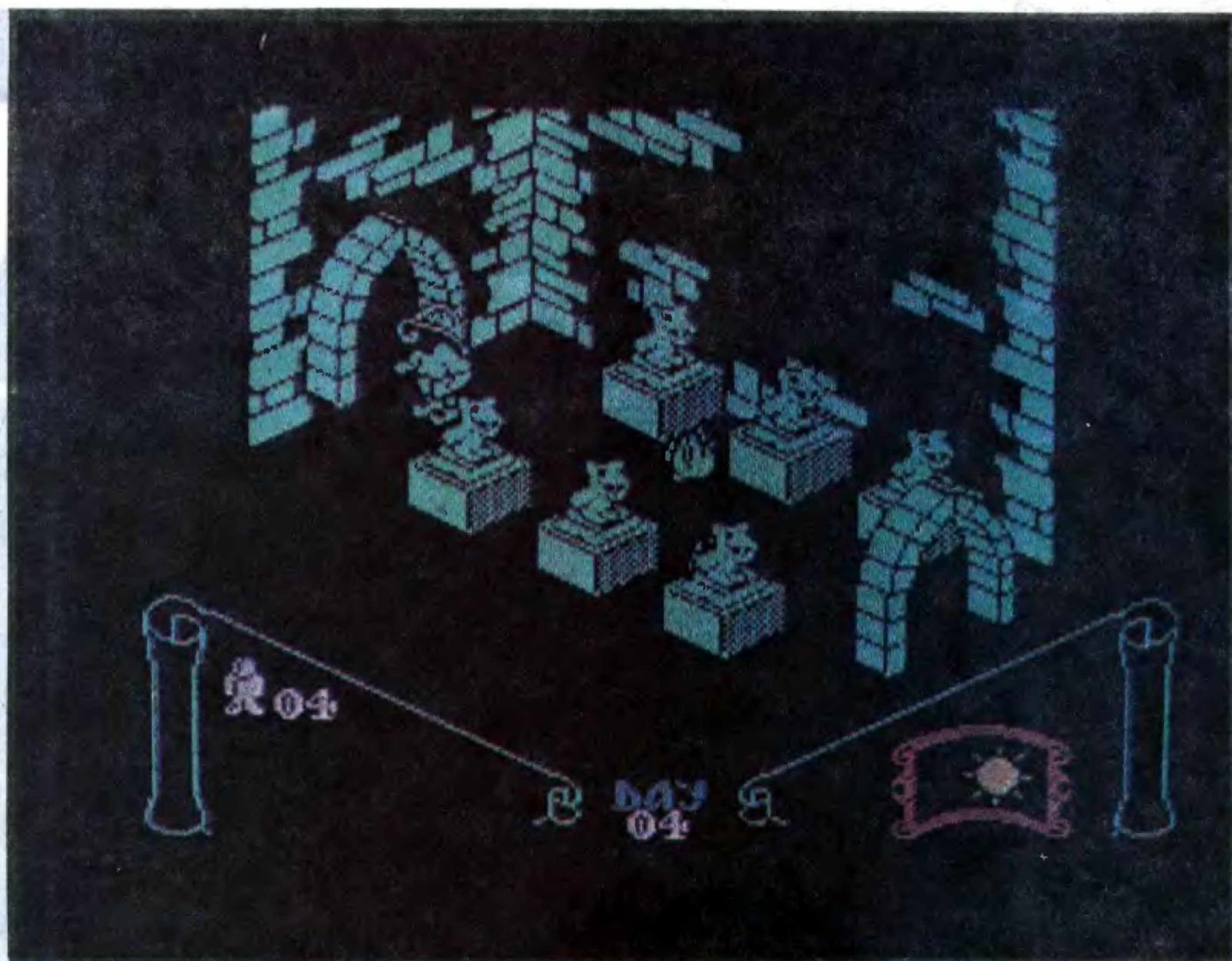


Inną wadą "Ortografii" jest jej zbyt mała – moim zdaniem – atrakcyjność. Zasada "uczyć – bawiąc" została przez J. Potempę uwzględniona w zbyt małym zakresie. Program jest bardzo "suchy", fachowy. A przecież wystarczy tak niewiele.

Pamiętam, dwa lata temu Piotr Parlewicz jako pierwszy podejmował próbę pisania mikrokomputerowej ortografii. Nie wiem nawet, czy praca została zakończona. Koncepcyjnie sprowadzała się do strzelania błędów. Na ekranie pojawiał się wyłącznie błędnie napisany wyraz. Zadaniem grającego było zniszczenie z laserowego działka (notabene pochodzącego wypisz wymaluj z programu "Galaxians") błędnie napisanej litery. Z punktu widzenia pedagogiki program był jeszcze gorszy, niż "Ortografia" J. Potempy. Ale za to ileż było przy nim zabawy!

I jeszcze jedna myśl, która nasunęła mi się na temat "Ortografii" – świadomość faktu, że zwrot "wieczór mickiewiczowski" napisany został poprawnie, a "dzieła mickiewiczowskie" – nie, wcale nie musi oznaczać równoczesnie świadomości tego, dlaczego w drugim przypadku mamy oddawać szacunek wieszczowi pisząc drugi wyraz zwrotu dużą literą, a w pierwszym nie. Odpowiedź "dlaczego" powinna być utrwalana na równi z właściwą pisownią określonego wyrazu czy zwrotu. Stąd potrzeba wyjaśnienia na ekranie monitora, jakiego rodzaju zasada została w danym przypadku naruszona. Myślę, że nawet przy rezygnacji z niektórych wyrazów (pamięć komputera nie jest wszak z gumy) przypomnienie czy wręcz uświadomienie tych zasad jest niezwykle potrzebne. W wielu przypadkach wystarczy bardzo skrótowe zasygnalizowanie, ot choćby "wymiana ó na o", "rz na r" czy wreszcie zwyczajny "wyjątek".

Marek Car



KNIGHT LORE

Gra, której mapę przedstawiamy na sąsiednich stronach, powstała w 1984 roku. Dwa lata to w świecie mikrokomputerów cała epoka. KNIGHT LORE dzielnie wytrzymuje próbę czasu i wydaje się być równie atrakcyjny jak w tydzień po ukazaniu się na rynku (chyba przede wszystkim dzięki znakomitej, dającej wrażenie trójwymiarowości obrazu grafice). Ci posiadacze ZX SPECTRUM, którzy zechcą poświęcić mu nieco czasu, na pewno nie wstaną od klawiatury zawiedzeni. Szczególnie teraz, gdy dzięki pismu KOMPUTER (to my!) nie będą musieli błądzić po labiryntach stanowiącego pole gry zamczyska po omacku.

Zadaniem gry jest uwolnienie Sabremana, dzielnego podróżnika, mającego upodobanie do za dużych, tropikalnych kapeluszy, od ciążącej na nim klątwy. W ciągu dnia wszystko jest w porządku – Sabreman jest zdrów i zadowolony, jednak gdy tylko zapadnie noc i na niebie pokaże się księżyc, klątwa daje o sobie znać. Nasz bohater zmienia się w straszliwego wilkołaka. Jeżeli my, przy klawiaturze czy dżojstiku, w ciągu 40 dni i nocy nie zdołamy mu pomóc, Sabreman na zawsze przybierze wilczą postać. A tego oczywiście nikt nie chce. Musimy więc poprowadzić go przez komnaty pełnego niebezpieczeństw zamku i odnaleźć czarownika Melkhiora, który, zwyczajem wszystkich magów, warzy w zaklętym kotle tajemnicze mikstury. Jedna z nich może uwolnić Sabremana od kłopotów. Niestety Melkhior jest już zbyt stary, aby samemu poszukiwać po zakamarkach zamku składników magicznego napoju – uprzejmie podpowiada jednak co mu jest potrzebne, by mikstura nabrała odpowiedniej mocy i właściwości. (Jednak uwaga – Melkhior nie znosi wilkołaków).

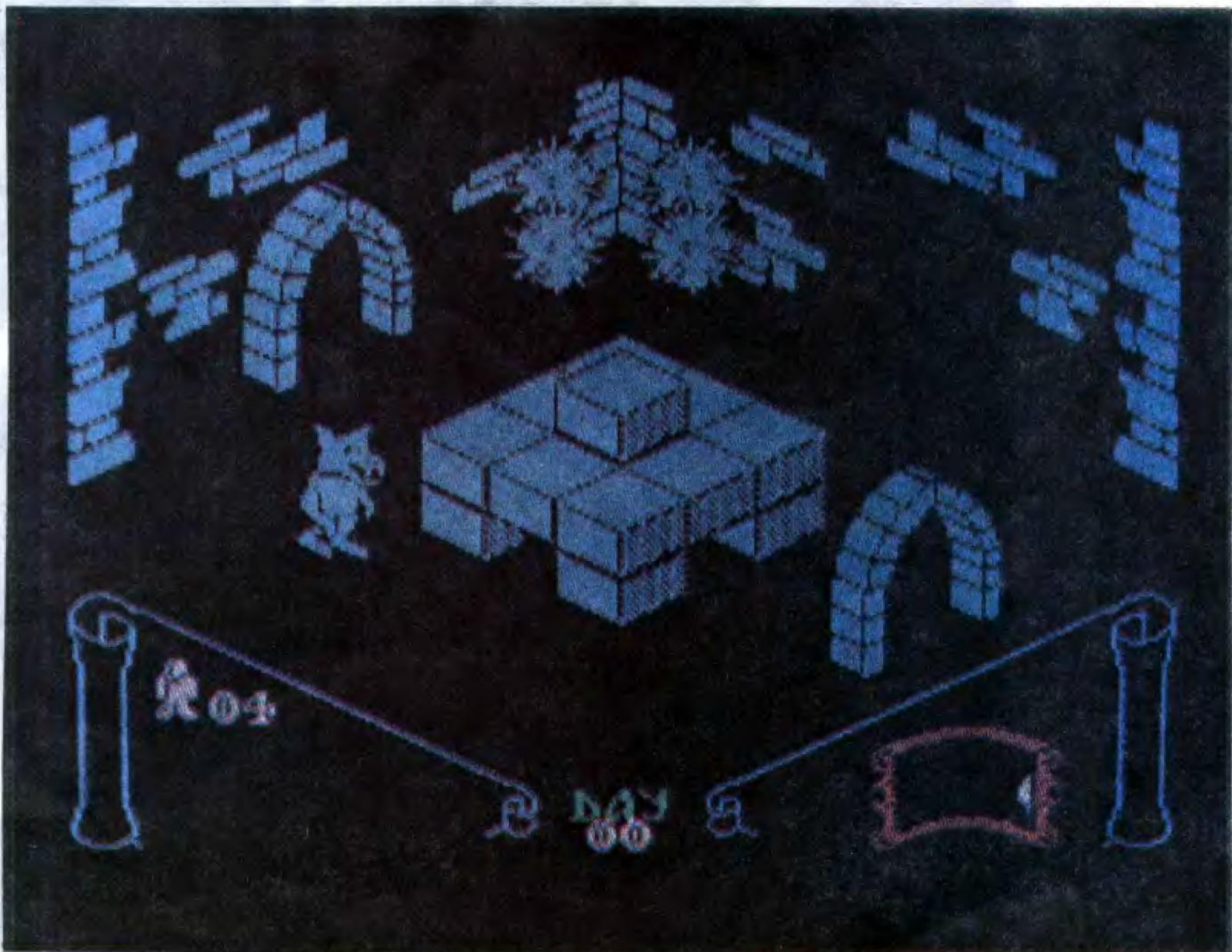
Tak więc Sabreman musi sam przeszukiwać labirynt kamiennych lochów i korytarzy, a to wcale nie jest łatwe. Przeszkody to nie tylko, a nawet nie przede wszystkim zawiłości typu architektonicznego. Prawdziwie niebezpieczni są mieszkańcy zamku oraz czyhające na każdym niemal kroku pułapki. Na życie naszego bohatera dybią uzbrojeni strażnicy, duchy potępionych rycerzy, a nawet pozornie niegroźne skaczące kule czy błędne ogniki... Wszędzie jeżą się zatrute ostrza, kolczaste kule spadają w najmniej oczekiwanych momentach, pozornie solidne, kamienne schody załamują się pod stopami lub wręcz rozsypują w pył... Taaak, ciężkie jest życie wilkołaka!

Część informacyjna ekranu (dół) zawiera odliczający czas wstecz kalendarz, symbol nieboskłonu z przesuwającym się słońcem lub księżycem, widok „kieszeni" Sabremana, a także ilość pozostałych mu jeszcze wskrzeszeń. Klawisze sterujące to: dolny rząd naprzemiennie – obrót w lewo lub w prawo; rząd następny – ruch w kierunku, w którym Sabreman jest zwrócony; powyżej – skok; i wreszcie klawisze najwyższego rzędu umożliwiają podnoszenie i wyrzucanie przedmiotów.

Dobrej zabawy!

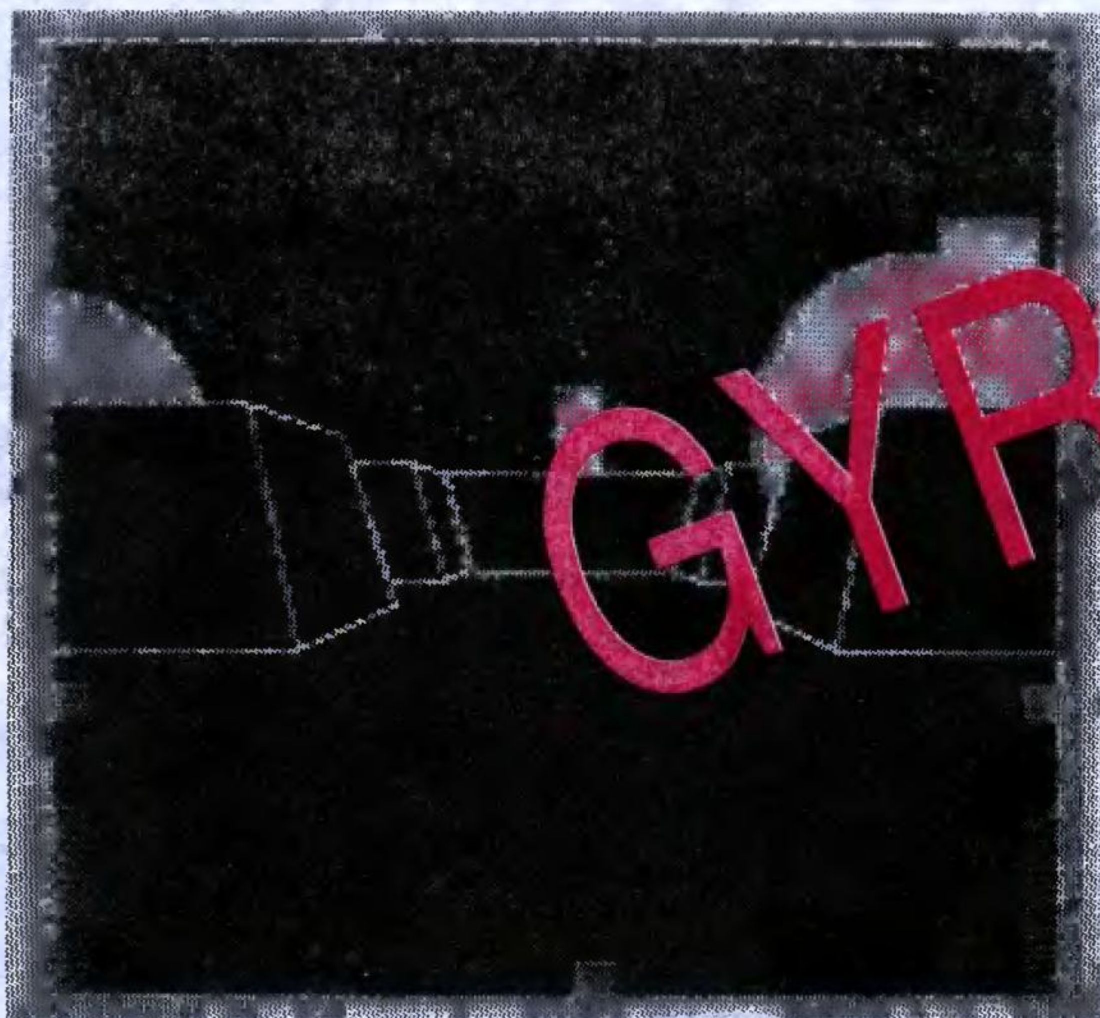
Mapa gry str. 24-25

(gc)



GRY





GYRON

Producent: FIREBIRD Komputer: ZX SPECTRUM 48K
Autorzy: Philip Mochan, Ricardo Pinto, Dominic Prior, Mark Wighton

Slogan reklamowy głosi, że GYRON to najtrudniejsza gra komputerowa, jaką kiedykolwiek napisano. To może być prawda, zwłaszcza sądząc po nagrodzie przyznawanej za jej ukończenie. To ni mniej ni więcej samochód Porsche 924! Wszystkich, którym w tym momencie zaświeciły się oczy, muszę jednak nieco ostudzić. Termin nadsyłania rozwiązań już minął i to dość dawno, bo 31 października 1985 r.

Na pierwszy rzut oka GYRON jest dość typową grą zręcznościową z rodzaju "wydostań się z tego labiryntu". Jednak po pewnym czasie okazuje się, że sprawność manualna jest przy jej rozgrywaniu umiejętnością co najmniej drugorzędną. W pewnym momencie gracz ma szansę ze zdziwieniem stwierdzić, że wymaga się od niego nie tylko szybkiego refleksu, ale również, o zgrozo!, logicznego myślenia.

Celem jest spenetrowanie labiryntu chroniącego wrogi komputer, dotarcie do tego komputera i zniszczenie go. Kompleks obronny został zaprojektowany w ten sposób, aby, będąc maksymalnie odpornym na zakusy najbardziej przemyślnych sabotażystów, umożliwiać jednak dostęp do komputera upoważnionym do tego osobom. Niestety gracz do tych osób się nie zalicza, musi więc sam, bez map i przewodników odnaleźć pozostawione przejścia. Nie osobiście rzecz jasna, lecz za pomocą zdalnie sterowanego, uzbrojonego w potężny laser robota. Przez lukę w ochronnym polu siłowym robot zostaje wprowadzony do labiryntu. A tam czyha na niego system obronny komputera i jego dwa wzajemnie się uzupełniające elementy: kule i wieże.

Kule są odporne na strzały z lasera i posiadają moc niszczącą wyzwalaną przy bezpośrednim kontakcie. Labirynt podzielony jest na rejony (można je odróżnić dzięki kolorowi wielościanów, obracających się na najniższym z "okien" po prawej stronie ekranu) i w każdym z tych rejonów kule poruszają się według ustalonego schematu. Przejście między nimi możliwe jest tylko określoną drogą i w ograniczonym czasie. Krążą pogłoski, że istnieją tylko cztery rozwiązania tego przestrzenno-czasowego dylematu.

Wieże, również utrudniające poruszanie się po labiryncie, mają zdolność rażenia na odległość, ustawiają się i strzelają w jednym z czterech głównych kierunków. Są jednak mniej odporne niż kule i można je niszczyć promieniami lasera. Są też głupsze – dają się przechytrzyć i ominąć przez przejście tuż koło muru, za którym wieża stoi lub przez wykorzystanie kuli jako osłony.

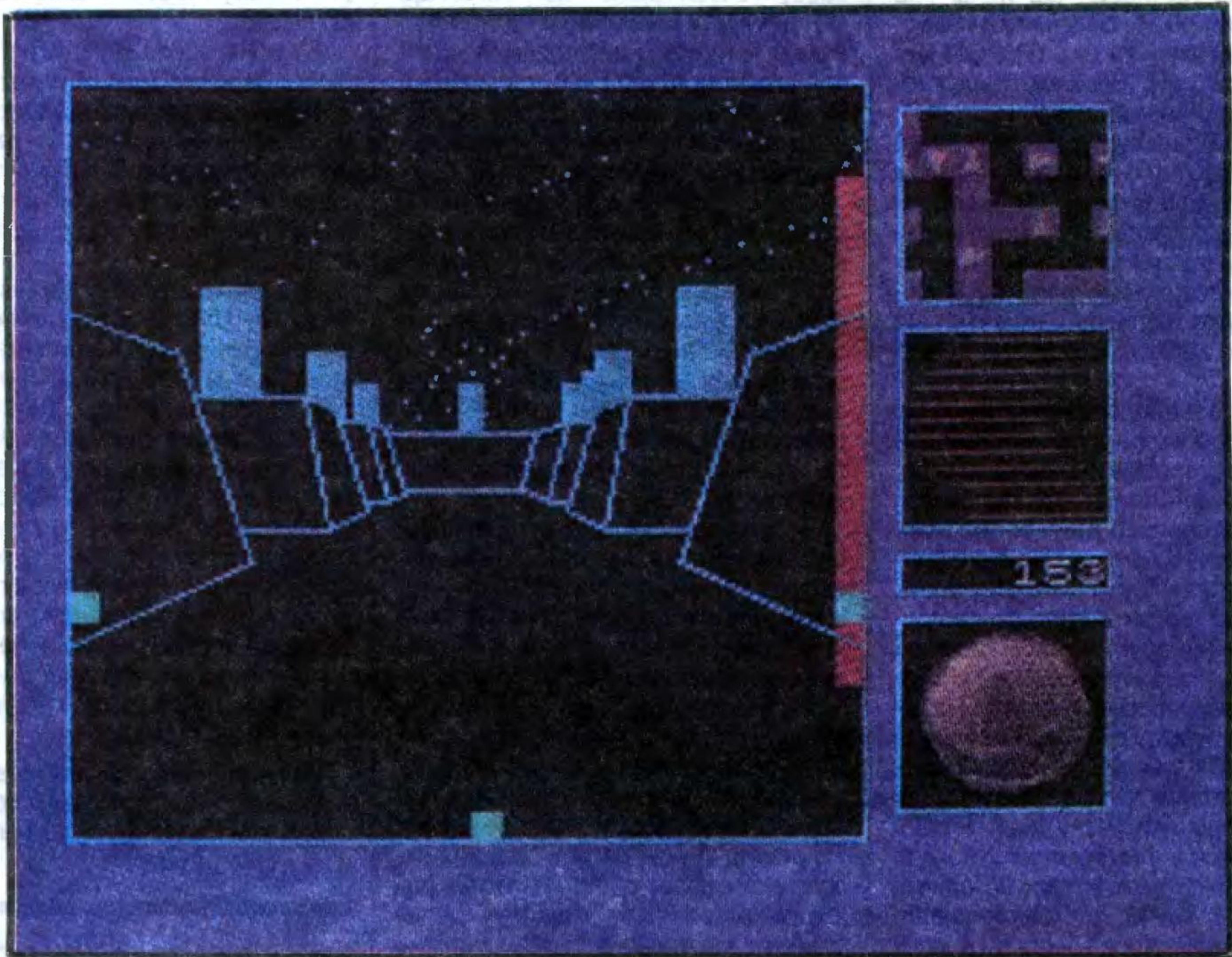
Główną część ekranu zajmuje duży, perspektywiczny obraz labiryntu, widzianego kamerami robota. Niewielki, zielony kwadrat przy jego dolnym skraju jest wskaźnikiem ruchu w lewo i prawo, zaś dwa takie same kwadraciki po bokach oznaczają pozycję robota w stosunku do bocznych korytarzy (czerwone paski). Skręt w boczny korytarz możliwy jest, gdy zielony kwadrat znajdzie się wewnątrz czerwonego paska. Po prawej stronie ekranu znajdują się cztery mniejsze "okna". Najwyżej położone przedstawia radarowy obraz okolicy z zaznaczonymi pozycjami kul i wież. Robot znajduje się zawsze w centrum. Poniżej falujące linie wskazują stopień zniszczenia robota – im jest ich mniej, tym zniszczenie większe. Trzecie od góry "okno" to zegar. I wreszcie najniżej znajduje się wirujący dwudziestościan (o ile nie pomyliłem się w obliczeniach) – swym kolorem odpowiada on rejonowi labiryntu, przez który robot aktualnie usiłuje się przedrzeć.

GYRON składa się z dwóch części, dwóch osobnych gier. Pierwsza z nich, ATRIUM, jest łatwiutkim wstępem, polem treningowym. Gracz ma dzięki niemu nabrać wyobrażenia o tym, co go czeka w czasie właściwej gry, czyli w części drugiej, zatytułowanej NEKROPOLIS.

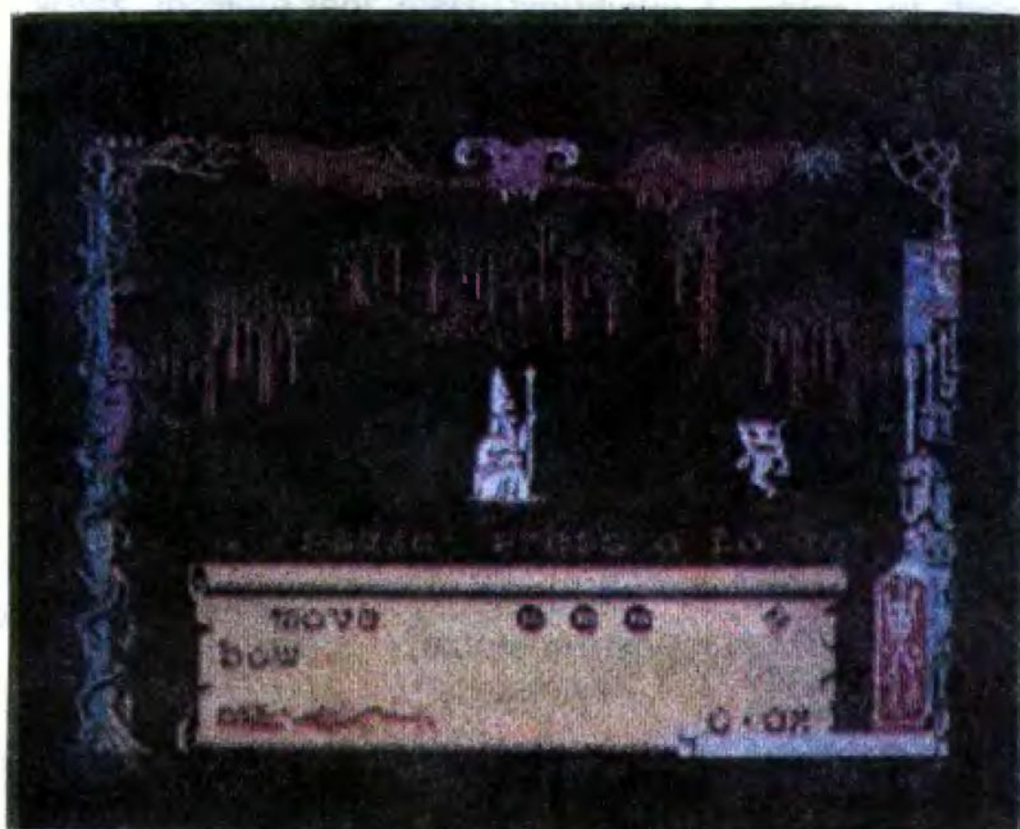
W sumie na pewno nie jest to gra, którą się kończy w pięć minut po jej wprowadzeniu do komputera – jeśli ktoś chce ją skończyć. Odnoszę jednak wrażenie, że teraz, gdy Porsche już dość dawno z wizgiem opon odjechał w siną dal, siła przyciągania GYRON-a znacznie spadła. Mimo niewątpliwych zalet nie jest to gra na tyle atrakcyjna, by trzymać w napięciu przez te godziny, dni, a może nawet tygodnie, które są potrzebne do jej rozwiązania. A poza tym Porsche i tak zawsze był poza naszym zasięgiem. Dzieliło nas kilka granic i przynajmniej jeden kanał.

GYRON kontrolowany jest przez większość systemów joystickowych, a z klawiatury za pomocą kursorów i klawisza jako "strzał".

GYRON



DRAGONTORC ■ DRAGONTORC ■ DRAGONTORC ■





DRAGONTORC

Wiele lat upłynęło odkąd Maroc zszedł w głąb Szklanej Góry, by odnaleźć i zniszczyć Bezimiennego, ponury cień, wiszący nad całym Avalonem. Wspomnienie tej strasznej wyprawy wciąż jeszcze mrozi mu duszę. Przez te wszystkie lata nie potrafił, jak niegdyś, siąść wśród ludzi, cieszyć się i śmiać jak oni. Nieustannie wędrował, stał się nomadem gęstych lasów, puszcz, w których ciągle jeszcze żyła magia.

Którejś nocy siedział, wpatrzony w żar wygasłego ogniska, słuchając opowieści wiekowych drzew, pamiętających dawne, zapomniane przez wszystkich wydarzenia. Nagle opanowała go dziwna niemoc. Poczuł dotknięcie czyjejś magii, czyjąś obecność... Jakiś głos z daleka, z bardzo daleka wołał jego imię... Nikt się nie pojawił, jednak głos, przyzywający, ciągle brzmiał w jego uszach.

Szedł przez wiele dni i nocy,śpiąc tylko wtedy, gdy zmęczenie zwalało go z nóg. Nie wiedział dokąd idzie ani co znajdzie na końcu tej wędrówki. Po prostu szedł, czując narastającą moc prowadzącej go magii. W końcu, na niewielkiej polanie, stanął przed starym, spróchniałym pniem dawno zwalonego drzewa. Zanurzył weń rękę. Poczuł pod palcami metal, niemal gorący, wibrujący magicznymi fluidami. Rozpoznał tę magię. Po chwili wpatrywał się w Pieczęć Merlina, ostatniego z Wielkich Magów, strażnika spuścizny Smoka. I znowu usłyszał głos. Tym razem silny i wyraźny, chociaż dobiegający z daleka. Głos swego dawnego nauczyciela i mistrza.

– Maroc, nareszcie... Jesteś jedynym, który może mi pomóc. Morag, ta wiedźma z północy, zwabiła mnie w pułapkę i uwięziła. Ale nie to jest najważniejsze – ja mogę poczekać. Przede wszystkim musisz udaremnić jej zamiary. Nie wystarcza jej to, co ma. Pragnie władzy nad wszystkim i wszystkimi. Chce zebrać pięć królewskich koron, tych, które wykuliśmy z Naszyjnika Smoka po odebraniu go Halgarowi. Jeśli jej się to uda, połączy je na nowo, odtworzy naszyjnik i posiądzie jego moc. A wtedy nikt się jej nie oprze. Musisz temu przeszkodzić. Czasu jest mało, już jedna z koron, króla Dumnovii, wpadła w jej ręce. Znajdź cztery pozostałe i zniszcz je. Potem przyjdź... – głos nagle osłabł i odpłynął, jakby ktoś usiłował go zagłuszyć, chwycić w sieci swych czarów i zdusić – przyjdź mnie uwolnić... Spiesz się...

Tyle mówi legenda. Dopisanie dalszego jej ciągu to już nasze zadanie. Na szczęście mamy komputer i program, który nam w tym pomoże.

Zanim zacznę sypać podpowiedziami i wyjaśniać, krok po kroku, całą grę od początku do końca, a taki właśnie mi zamiar przyświeca, kilka słów na temat "jak w to grać?". Ci czytelnicy, którzy zetknęli się z programem LEGEND OF AVALON lub posiadają instrukcję do niego, notabene krążącą od pewnego czasu w polskim tłumaczeniu, mogą ze spokojnym sumieniem ominąć kilka następnych akapitów i, jeśli im to odpowiada, przejść od razu do odpowiedzi. I tak już wszystko wiedzą.

Program DRAGONTORC jest typowym przykładem łączenia gry zręcznościowej, wymagającej dobrego refleksu i sprawnych w operowaniu klawiszami palców z przygodową, a więc polegającą na rozwiązywaniu różnorodnych zagadek i problemów. Typowym przykładem, ale to nie znaczy, że mało oryginalną grą. DRAGONTORC na pewno potrafi wzbudzić zainteresowanie a nawet zafascynować. Mam nadzieję, że czytelnicy sami się o tym przekonają.

Sterowanie ruchem bohatera gry jest dosyć proste. Może on się poruszać w górę, w dół, w lewo lub w prawo (np. odpowiednio klawisze Q, A, N, M). Problemy pojawiają się dopiero przy używaniu czarów. Po naciśnięciu klawiszy K lub L ("strzał", jeżeli posługujemy się joystickiem) na "pergaminie" na dole ekranu pojawia się spis posiadanych przez Maroca magicznych zaklęć. Używając klawiszy "dół" lub "góra" należy ustawić pożądane zaklęcie na wprost strzałki, a następnie nacisnąć K lub L. Czar jest przygotowany do użycia. Ponownie K lub L... no i o to właśnie chodziło. Czary dzielą się na cztery rodzaje. Dwa z nich (np. Bane i Servant) działają stale, aż do wyłączenia, które może (chociaż nie musi) nastąpić identycznie jak uruchomienie. Pozostałe (np. Energize i Missile) wyłączają się automatycznie natychmiast po użyciu. I jeszcze jedno – czary należy oszczędzać, w pewnym momencie mogą się wyczerpać.

Proszę mi wybaczyć częste używanie oryginalnych, angielskich nazw czarów, niektórych przedmiotów czy miejsc. Tłumaczenie ich bowiem na polski czasami utrudniałoby, lub wręcz uniemożliwiało ich rozpoznanie w trakcie gry.

A teraz kolej na "psucie" zabawy, czyli podpowiedzi.

Maroc rozpoczyna grę na polance w Wispwood Forest. Posiada trzy zaklęcia: Move, Bane i Servant. Na Move można w ogóle nie zwracać uwagi. Bane to magiczna tarcza, chroniąca przed leśnymi duchami, upiorami i oczami (tak, tak – przed złymi, latającymi oczami). Servant to duszek-sługa, dzięki któremu Maroc może podnosić przedmioty, chować je do plecaka czy też posługiwać się nimi. Może również otwierać lub zamykać drzwi. Steruje się nim za pomocą tych samych klawiszy, dzięki którym porusza się Maroc. Servant może zostać wyłączony poprzez wyprowadzenie duszka poza skraj ekranu.

A więc do dzieła. Za pomocą czaru Bane pozbądź się czterech leśnych duchów. Bądź ostrożny, gdyż Bane zużywa energię (czerwony płomień na dole ekranu). Weź Merlyn Seal (Pieczęć Merlina), leżącą koło ogniska i z pomocą Servant przesuń kamienne płyty. Spod jednej z nich wyjmij znajdujący się tam przedmiot, spod drugiej czar Missile (magiczne pociski). Znajdujący się w stawie Jewel (brylant) oraz łuk i strzały daj elfom. W zamian otrzymasz wiadomość oraz sierp. Zetnij i weź roślinę. Uzyskasz w ten sposób Energize, czar uzupełniający energię. Wróć do otworu, odsłoniętego przez jedną z kamiennych płyt i stań na nim. Znajdziesz się w Lost Vaults of Locris.

Idź na prawo do pokoju, w którym znajduje się dzbanek do herbaty. Przeszukaj wszystko i weź Rune (znak runiczny). Teraz idź na lewo, do pokoju z ognistą kulą. Dotknij stołka (Servant) i zmieni się on w skrzynię. Wyjdź przez drugie drzwi. Ucieknij kościotrupowi i przejdź dalej, do następnej komnaty. Zabierz czar i idź naprzód. Pozbądź się kościotrupa przy pomocy Missile i przejdź przez drzwi, przyległe do tych, którymi tu wszedłeś. Weź CHALICE (kielich). Wróć do komnaty, w której był staw i zanurz w nim kielich. Wróć do pokoju z ognistą kulą, a stamtąd idź dwa razy na prawo. Powinieneś znaleźć się w pomieszczeniu, w którym znajduje się coś nieokreślonego, przypominającego duży otwieracz do konserw. Wylej na to zawartość kielicha – otrzymasz Heal, czar leczący ślepotę i ukąszenia węży. Usiądź na otwieraczu. Znajdź trzy pokoje ze skrzyniami. Uważaj na pająki. Jedną z tych skrzyń może otworzyć Servant. Znajduje się w niej klucz do drugiej, a w tej z kolei klucz do trzeciej. Po jej otworzeniu weź Leyrod oraz klucz. Za pomocą tego klucza otwórz skrzynię w pokoju z ognistą kulą (tę, która przedtem była stołkiem). Wyjmij ze skrzyni kolejny znak runiczny. Zabierz również Halfmoon (półksiężyc), w który zmieniła się ognista kula. Idź do komnaty, w której znajduje się znak runiczny E. Zostaw tu wszystkie znalezione do tej pory przedmioty. Odszukaj dwa leżące na podłogach szkielety. W jednym z nich ukryty jest ostatni znak runiczny, zaś w drugim czar Missile oraz niespodzianka. Idź do pokoju, przyległego do tego, w którym zaczynałeś pobyt w Vaults of Locris. Złap przy pomocy Servanta latający czar. Wróć do pomieszczenia, w którym zostawiłeś przedmioty. Zabierz je i ułóż na znaku E pozostałe runy, w kolejności X – I – T. Ukaże się wirujący sześcian. Wejdź na niego. Przeszukaj Wierdhenge, kamienny krąg w lesie Wispwood za pomocą czaru Leyrod. Ukażą się dwa wirujące sześciany. Wejdź na żółty.

Na razie dosyć. Dalszy ciąg w następnym numerze.

Powodzenia!

Sufler

OGŁOSZENIA



ZX SPECTRUM SERVICE, PMS ELEKTRONIK

ul. Legionowa 23, 01-343 Warszawa, skr. poczt. 17
Poleca:
– serwis komputerów firmy Sinclair Research Ltd.,
– interfejsy do joysticków systemu Kempston,
– interfejsy do drukarek systemu Centronics,
– rozszerzanie pamięci RAM,
– kable monitorowe, TV oraz inne,
– programy użytkowe oraz gry komputerowe dla komputera ZX Spectrum

BR-139

- programy użytkowe
- konsultacje
- programy specjalistyczne na zamówienie
- oprogramowanie systemów dyskowych, interfejsów i drukarek

oferuje:

Studio Programów Mikrokomputerowych „SPECTRUM"

mgr inż. Tadeusz Wilczek
koresp.: 03-562 Warszawa, ul. Gajkowicza 3 m 26
tel. 15-38-58 środy $13^{00} - 18^{00}$

BR-143

POLKONAR
61-895 Poznań
tel. 568-00, godz. 8^{00}-15^{00}
ul. Powstańców Wielkopolskich 14/7

Naprawa mikrokomputerów ZX 81, ZX Spectrum, Commodore. Wyrób i sprzedaż interfejsów; typy: do stacji dysków (tylko do ZX Spectrum), standard, interfejsy z przetwornikami A/C, C/A (do 12 bitów), interfejsy z wejściami i wyjściami dowolnie programowanymi, interfejsy do drukarek (równoległe i szeregowe), interfejsy do stacji papierowych, interfejsy do sprzęgania z innymi komputerami, zabezpieczenia pamięci ZX Spectrum.
Firmom wystawiamy faktury.

BR-142

SPECTRUM
COMMODORE 64
AMSTRAD

Programy. Instrukcje. Literatura.
DH „Sezam" IIp. g.16-19

BR-138

STUDIO KOMPUTEROWE
PROGRAMY – INSTRUKCJE – LITERATURA
HALA MIROWSKA
godz. 11-19 (także w soboty)

BR-140

Firma MUEL oferuje do sprzedaży:
1) INTERFEJS do ZX SPECTRUM, umożliwiający współpracę z czterema napędami dysków elastycznych dowolną drukarką graficzną, monitorem ekranowym, rozszerzający BASIC oraz system operacyjny ZX SPECTRUM. Nie zajmuje pamięci RAM!
2) Sterowany „ikonami" programator EPROM 2716÷27256 do ZX SPECTRUM
3) Przeróbkę drukarki DZM 180 na drukarkę graficzną.
Informacja tel.: 33-40-91
Korespondencja: MUEL
ul. Cząstkowska 30
01-678 Warszawa

BR-141



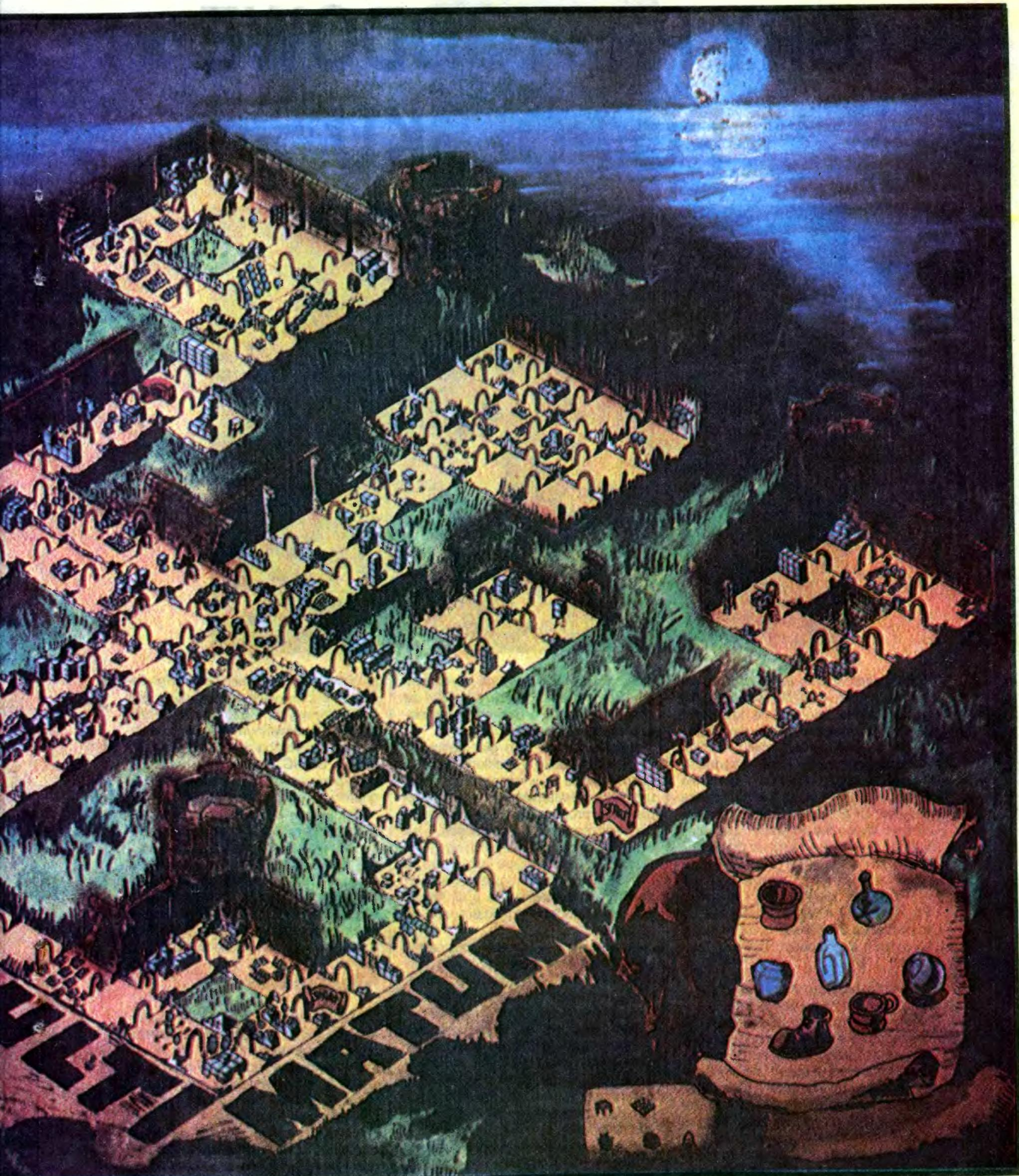
Ogłoszenia i korespondencję w sprawach ogłoszeń przyjmuje Krajowe Wydawnictwo Czasopism, ul. Noakowskiego 14, 00-666 Warszawa (adres dla korespondencji). Dział reklamy KWCz mieści się przy ul. Mokotowskiej 5 (tel. 25-35-36).

Opłaty za ogłoszenia (od osób prywatnych płatne z góry) należy wpłacać w kasie Wydawnictwa lub na konto w NBP III O/Warszawa nr 1036-5294.

1 cm^2 kosztuje 300 zł,
najmniejsze ogłoszenie drobne (7 cm^2, czyli 3 wiersze albo ok. 20 słów) kosztuje 2100,-
za ogłoszenia z 1 kolorem dodatkowym dolicza się 30%, natomiast za ogłoszenia w pełnej gamie barwnej – 100%.
Odpowiednio droższe są ogłoszenia na ostatniej stronie okładki.
Dla zamawiających większe serie ogłoszeń oraz dla firm umożliwiających redakcji testowanie ogłaszanego sprzętu i produktów przewidziane są ulgi i inne udogodnienia.
Uwaga! Osoby wpłacające pieniądze na konto proszone są o zaznaczenie na blankiecie przekazu, że opłata jest należnością za ogłoszenie w „Komputerze". Treść ogłoszenia można przekazać listownie podając datę i miejsce wpłaty.

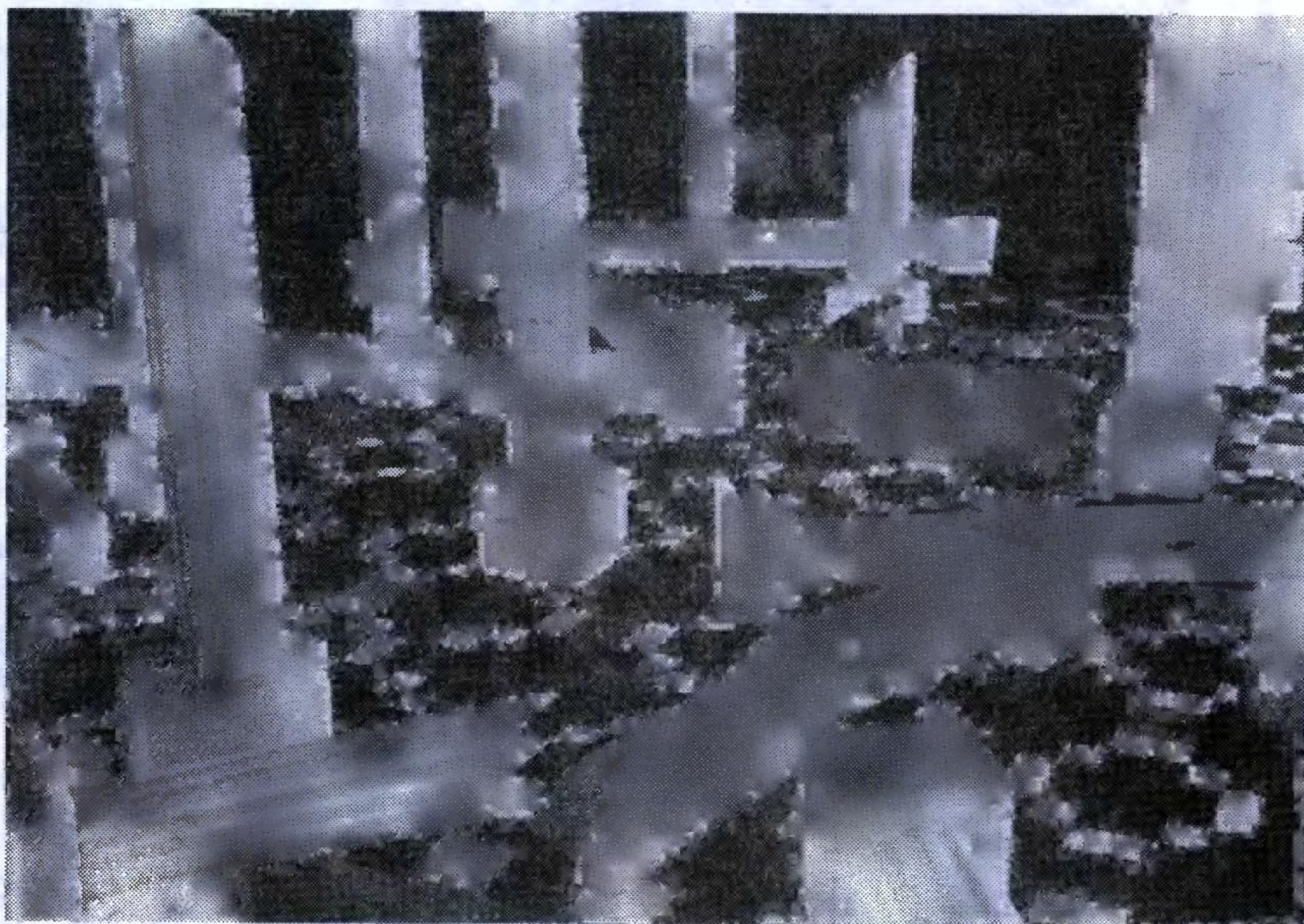
KNIGHT LORE





Patrz opis gry na str. 17

GRY PRZYGODOWE PO POLSKU



Po dwóch dniach wspinaczki dotarłeś wreszcie do celu. Na końcu skalnej półki widać ciemne wejście do pieczary. Zastanawiasz się, jakie niebezpieczeństwa czyhają na Ciebie w mroku, z jakimi tajemnymi mocami będziesz musiał walczyć. Wierzysz, że Twoja odwaga, siła i Twój niezawodny miecz pozwolą Ci pokonać Demona Nocy i zdobyć Magiczny Kryształ. Zapalasz lampę i ostrożnie wchodzisz do jaskini. Na twarzy osiadają Ci pajęczyny. Słyszysz ciche szmery. To pewnie szczury. Powietrze jest chłodne i wilgotne. Po kilkunastu metrach docierasz do rozwidlenia. Korytarze prowadzą na wschód i na zachód. Który wybierzesz?

Tak mogłaby się zaczynać gra przygodowo-tekstowa... gdyby ktoś ją napisał po polsku. Programy takie już dawno zdobyły sobie wielką popularność na rynkach zachodnich i z powodzeniem konkurują z grami zręcznościowymi. Wielu uważa je za wyższy etap wtajemniczenia po ogłupiających zabawach typu pif-paf! bij-zabij! W tych przygodach laików najbardziej zdumiewa zdolność komputera do porozumiewania się z grającymi w języku naturalnym. To jednocześnie... główna bariera ograniczająca ich powszechność wśród polskich użytkowników komputerów domowych, jako że znajomość języków obcych nie jest naszą najmocniejszą stroną.

Polskie wersje tych gier pojawiają się bardzo rzadko. Przekłady takie są pracochłonne i wymagają na ogół umiejętności włamywania się w zabezpieczone programy oraz modyfikowania szczegółów technicznych, jak choćby wprowadzania polskiego liternictwa. Zależnie od typu komputera są to problemy o różnej złożoności i w dodatku bardzo zawiłe. Na ogół łatwiej jest samemu napisać taki program od początku.

Istnieje wiele schematów tworzenia gier przygodowo-tekstowych, od bardzo prostych po niezwykle wyrafinowane. Powstały programy pozwalające każdemu względnie szybko i łatwo konstruować własne oryginalne fabuły, uwalniające od troski o szczegóły techniczne realizacji. Ich wspólną wadą jest jednak to, że dostosowane do potrzeb języka angielskiego – nie dają się bezpośrednio stosować w innych językach.

Jednakże nawet najprostsze techniki, korzystające jedynie z języka BASIC, pozwalają na pisanie bardzo rozbudowanych i ciekawych gier.

Poniżej opiszemy jeden z takich schematów. Będzie to nieco uproszczona wersja techniki z powodzeniem stosowanej przez niektóre firmy profesjonalne zajmujące się pisaniem gier komputerowych. Może ona być realizowana na dowolnym komputerze domowym. Wymaga jednak znajomości języka BASIC we własnej maszynie oraz pewnej wprawy w pisaniu i uruchamianiu dłuższych programów.

W naszym opisie ograniczymy się do podania ogólnej metody konstrukcji, bez wnikania w szczegóły konkretnej realizacji na danej maszynie. W nielicznych przykładach będziemy posługiwali się instrukcjami BASICU w wersji ZX SPECTRUM. Użytkownicy innego sprzętu z łatwością będą mogli dostosować je do własnych możliwości.

Miejsce i czas

Najtrudniejsze i najbardziej pracochłonne jest szczegółowe opracowanie fabuły atrakcyjnej gry. Na dobre pomysły nie ma recepty. W początkowej fazie pracy jesteśmy zdani całkowicie na własną pomysłowość, fantazję i poczucie humoru.

Najpierw trzeba sporządzić mapę obszaru, na którym będzie się toczyć gra. Powinna to być ponumerowana lista wszystkich miejsc, w których planujemy się znaleźć, wraz z ich opisem słownym. Na przykład miejscem numer 1 może być wejście do jaskini opisane tak, jak na wstępie artykułu. Opis

powinien przedstawiać dane otoczenie, informować co się w nim znajduje i w jakich kierunkach można się z niego przemieszczać (o ile nasz scenariusz będzie taką możliwość przewidywał).

W komputerze mapę przedstawimy jako tablicę M (n,4). Liczba n będzie równa liczbie miejsc na mapie, zaś 4-ka to cztery strony świata. Przy mapie wielopoziomowej należy raczej użyć tablicy o wymiarach M (n,6), tak, by uwzględnić również kierunki DÓŁ i GÓRA. Wartość M (i,j) to numer pomieszczenia, do którego można przejść z i-tego miejsca idąc w j-tym kierunku. Jeśli z naszego rozwidlenia korytarze prowadzą odpowiednio do 3-go pomieszczenia, w 2-gim kierunku (wschód) lub 7-go w 4-tym (zachód), to pierwszy wiersz tablicy M będzie miał postać: 0,3,0,7,0,0. Zera sygnalizują niemożność bezpośredniego posuwania się w żądanym kierunku. Ważne jest, by w tablicy M były zaznaczone tylko te przejścia, pokonanie których nie wymaga spełnienia żadnych dodatkowych warunków. Jeśli więc między pokojem numer 7 a 8 na wschodniej ścianie są zamknięte drzwi, to M (7,2) = 0, a nie 8, gdyż te drzwi trzeba będzie uprzednio otworzyć (być może przy pomocy klucza, magicznego zaklęcia, laski dynamitu albo jeszcze czegoś innego).

Przedmioty

Następnie sporządzamy listę wszystkich przedmiotów, którymi grający może się posługiwać, to znaczy przemieszczać je, zabierać ze sobą i wykorzystywać w innych miejscach.

Wszystkie te nazwy umieścimy we wspólnej tablicy P$ (p,20), gdzie p oznacza ilość przedmiotów, zaś 20 jest długością maksymalnego opisu. Na przykład w naszej tablicy mogą się znaleźć: P$ (1) = "Pudełko zapałek".
P$ (2) = "20 metrów lontu".
P$ (3) = "Dynamit".
P$ (4) = "Stary pergamin".
P$ (5) = "Złoty kluczyk".
P$ (6) = "Pusty rewolwer".
P$ (7) = "Nabity rewolwer".
P$ (8) = "6 naboi rewolwerowych".

Niektóre przedmioty występują w dwóch różnych stanach. Np. rewolwer może być nabity lub nie. Są one wymienione na liście dwukrotnie i obok siebie, jakby każdy możliwy stan opisywał zupełnie inny przedmiot.

Jednocześnie z tablicą P$ będziemy używać tablicy G (p), by pamiętać gdzie w danym momencie znajduje się dany przedmiot. G (3) = 7 oznacza, że dynamit leży sobie na wierzchu w 7-mym pomieszczeniu. Wartością 0 będziemy sygnalizować, że dany przedmiot mamy przy sobie, zaś wartościami ujemnymi oznaczymy te, które są niedostępne lub aktualnie niewidoczne (pistolet nie może być jednocześnie nabity i nie nabity, a kluczyk może być ukryty w zamkniętej szkatule).

Warunki

Kolejny etap to sporządzenie listy wszystkich warunków mogących wpływać na wyniki podejmowanych akcji. Na tej liście nie umieszczamy warunków typu: czy posiadasz dany przedmiot, bo możemy to sprawdzić badając odpowiedni element tablicy G.

Warunki te zostaną ujęte w tablicy W (w), gdzie w oznacza ilość możliwych testów. Przypuśćmy, że drzwi pomiędzy pomieszczeniem 7 i 8 można otworzyć przy pomocy magicznego zaklęcia. Jest ono zawarte w starym pergaminie, ukrytym w kufrze w miejscu numer 17. Przy komendzie OTWÓRZ DRZWI trzeba przetestować warunek W (1), który powie czy gracz zna już zaklęcie (1), czy jeszcze nie (0), a więc czy przeczytał stary pergamin. Ale jeszcze przed odczytaniem starego pergaminu trzeba będzie sprawdzić warunek W (2), ustalający czy skrzynia została otwarta, a jej zawartość stała się dostępna dla grającego. Po otwarciu drzwi z kolei zmienimy wartość W (3) z 0 na 1, sygnalizując w ten sposób, że drzwi zostały otwarte i można iść na wschód.

Jeśli nasz scenariusz przewiduje, że pewne wielkości w czasie gry mogą ulegać zmianom (np. ubywa amunicji w czasie strzelaniny, gracz traci siły w walce wręcz, a odzyskuje je podczas posiłków lub snu), to trzeba dla nich zarezerwować tablicę Z (z). Poszczególne wartości w tej tablicy poinformują o zapasie poszczególnych wielkości (sił, amunicji itp.).

Mając już wykaz warunków i rozmiary zapasów, zaprojektujmy serię możliwie prostych procedur sprawdzających je. Zarezerwujmy zmienne N, MIEJSCE, DOBRZE do następujących celów: N-parametr przekazywany procedurze precyzujący co ona ma robić w danym wywołaniu. MIEJSCE-zmienna pamiętająca, gdzie się aktualnie znajduje gracz. DOBRZE-zmienna, przy pomocy której będzie zwracana odpowiedź (0 warunek niespełniony, 1 spełniony).

Następne procedury oznaczymy kolejnymi dużymi literami alfabetu. Na pewno przydadzą się poniższe testy:

A – LET DOBRZE = (N= MIEJSCE OR N = 0) : RETURN

Procedura ta odpowie 1, jeśli znajdujemy się we właściwym miejscu, bądź miejsce nie ma znaczenia (N=0).

B – LET DOBRZE = (P(N) = MIEJSCE OR P(N) = 0) : RETURN

Tu odpowiedź określi, czy n-ty przedmiot znajduje się w miejscu, w którym właśnie jesteśmy, czy już je minęliśmy.

C – LET DOBRZE = (W(N) = 1) : RETURN

D – LET DOBRZE = (W(N) = 0) : RETURN

Powyższe procedury odpowiedzą na pytanie, czy n-ty warunek jest spełniony czy nie.

E – LET DOBRZE = (Z(N) =0) : RETURN

Tu z kolei możemy otrzymać potwierdzenie posiadania zapasu danych dóbr, np. czy w rewolwerze mamy jeszcze naboje.

G – LET DOBRZE=(N = MIEJSCE) : RETURN

Wartość DOBRZE = 1 sygnalizuje naszą nieobecność w n-tym pomieszczeniu.

Wygodnie jest zaprojektować procedury w taki sposób, by po odpowiednio postawionym pytaniu otrzymywać odpowiedź twierdzącą. Dlatego wypisaliśmy zarówno procedurę C jak i D, mimo że w zasadzie obie informują o tym samym. W dalszym ciągu powinno samo się wyjaśnić, dlaczego właśnie takie postępowanie ułatwi nam grę.

Zmiany

Podobnie, oznaczając dużymi literami alfabetu, musimy zaprojektować zestaw możliwie krótkich podprogramów określających dopuszczalne zmiany w otoczeniu, czyli w praktyce modyfikujących nasze tablice stosownie do rozwoju sytuacji. Przykładowymi procedurami tego typu mogą być:

```
A – PRINT "Niesiesz ze sobą:"
FOR I = 1 TO P
IF P(I) = 0 THEN PRINT P$ (I)
NEXT I
RETURN
B – LET S(N) = 1 : RETURN
C – LET S(N) = 0 :RETURN
D – LET Z(N) = Z(N) – 1 : RETURN
E – LET POM = G(N) : LET G(N) =
G(N + 1) : LET G(N + 1) = POM :
RETURN
```

W podprogramie E zmienna POM pełni jedynie rolę zmiennej pomocniczej. Faktycznie bowiem dokonuje się zamiana miejscami wartości G(N) i G(N + 1). Rozpatrzmy efekt wywołania tej procedury z parametrem N = 6. W naszych przykładach oznaczało to "NABITY REWOLWER". Jeśli pierwotna wartość G(6) była 0, to teraz stanie się ujemna. Okaże się więc, że po prostu wystrzelaliśmy już wszystkie naboje i nasza broń chwilowo może służyć jedynie jako podręczny młotek.

F – RESTORE 6000 + 10 x N : READ A$: PRINT A$: RETURN

Jeśli począwszy od linii 6000 umieścimy instrukcje DATA "tekst", to powyższa procedura odda nam nieocenione usługi przy drukowaniu na ekranie odpowiednich napisów. Wśród nich umieścimy bowiem wszystkie opisy pomieszczeń, wyniki działań gracza, teksty wskazówek, czyli praktycznie niemal wszystko. Tego rodzaju wykorzystanie instrukcji DATA zamiast tablicy znakowej ma tę zaletę, że pozwala w istotny sposób oszczędzać pamięć, szczególnie w sytuacji, gdy poszczególne komunikaty istotnie różnią się od siebie długością.

Dalsze procedury będą spełniać podobne funkcje zależnie od potrzeb naszej fabuły. Dbajmy jedynie o to, by były krótkie. Najlepiej jeśli każda z osobna zajmie nie więcej niż 9 linii. Do komputera wprowadzimy je potem począwszy od linii 8000 tak, by każda z nich mogła być wywołana instrukcją GO SUB 8000 +10 x (CODE "A" – CODE L$), gdzie L$ jest literą oznaczającą daną procedurę. (Ciąg dalszy w następnym numerze).

Andrzej Kadlof

ZX SPECTRUM 128K

W końcu 1985 r. labolatorium badawcze firmy INVESTRONICA mieszczące się w Madrycie, przedstawiło nową wersję komputera ZX SPECTRUM model 128K. Firma ta jest dystrybutorem wyrobów SINCLAIRA na rynku hiszpańskim. Wkrótce ma uruchomić własną produkcję ZX SPECTRUM+. Nowy model opracowano wspólnie z SINCLAIREM. Ma być on przeznaczony na rynki Hiszpanii i Meksyku. Sądzimy, że czytelników zainteresują szczegóły dotyczące tej konstrukcji.

Zewnętrznie model 128K jest bardzo podobny do SPECTRUM+: ma identyczną klawiaturę. Z prawej strony, na zewnątrz wyprowadzony jest spory radiator do odprowadzania ciepła (którego odprowadza niemało, bo przy dotknięciu wręcz parzy!). Po lewej stronie umieszczono wyjścia słuchawkowe i mikrofonowe oraz połączone wyjście wbudowanych interfejsów RS 232 oraz MIDI. Z tyłu, oprócz standardowego złącza i gniazda zasilania, są wyjścia video oraz RGB do współpracy z monitorem kolorowym. Niespodzianką jest czterodziałaniowy kalkulator, podłączony z przodu osobnym kablem. W czasie pracy z edytorem ekranowym jego klawisze sterują kursorem.

Ilekroć rozpatruje się nową wersję jakiegoś komputera, zawsze pojawia się pytanie o zgodność programową (kompatybilność) z poprzednią wersją. W przypadku SPECTRUM jest to problem szczególnie istotny z uwagi na bogactwo dostępnego oprogramowania.

W SPECTRUM 128K konstruktorzy rozwiązali to zagadnienie bardzo radykalnie. Po wydaniu rozkazu "SPECTRUM" (osobny klawisz) odłącza się RS232, MIDI, kalkulator i przed użytkownikiem leży SPECTRUM 48K. Do trybu 128K, wykorzystania dodatkowej pamięci czy wbudowanych interfejsów nie można wrócić nawet z kodu maszynowego. Zapewnia to całkowitą zgodność programową z, jak na razie, małym ale. Pierwsze egzemplarze, w miejsce pamięci typu ROM, otrzymają elementy typu EPROM. W rezultacie programy wykorzystujące przebiegi czasowe procedur z ROMU (np. niektóre tzw. speed loader'y), na ogół nie działają zgodnie z oczekiwaniami. W przyszłości jednak SPECTRUM 128K ma być wyposażony w tradycyjne pamięci typu ROM.

Dopiero po wciśnięciu klawisza RESET (chwilowe odłączenie napięcia zasilającego) komputer wraca do trybu 128K. Mamy w nim do dyspozycji rozbudowany edytor ekranowy. Zmiany w programach można dokonywać na całym ekranie. Większość komend sterujących kursorem wymaga stosowania dodatkowych klawiszy kalkulatora. Pozwalają one między innymi na przesuwanie kursora o jeden znak, słowo, przejście na początek poprzedniej linii lub koniec następnej. Usuwać można zarówno pojedyncze znaki jak i całe słowa i to w dowolnym kierunku. Edytor potrafi również złożone linie wypisywać na ekranie tak, że w jednej linii znajduje się tylko jedna instrukcja. Zawartość ekranu można przesuwać zarówno w górę jak i w dół. Nie przewidziano kopiowania czy usuwania grupy linii, automatycznego ich numerowania ani możliwości przenumerowania wszystkich linii programu, chociaż te dwie ostatnie czynności mają w przyszłości być wprowadzone. Przejście od pracy z edytorem do trybu wykonywania komend i na odwrót, wymaga specjalnych rozkazów.

W modelu tym zrezygnowano z możliwości wprowadzania całych słów języka BASIC przy pomocy pojedynczych klawiszy. Nie są również akceptowane żadne skróty. Sam język jest w zasadzie ten sam co w klasycznej wersji SPECTRUM. Rozszerzono go jedynie o rozkazy sterujące dostępem do pamięci i dźwiękiem. Możliwe jest pisanie programów w BASICU z pełnym wykorzystaniem możliwości edytora, a następnie przełączenie się na 48K. Programy nie korzystające z trików i nie mieszające w zmiennych systemowych będą działały w obu trybach.

Ponadto edytor może być również, w ograniczonym zakresie, używany jako edytor tekstowy. Możliwa jest bowiem edycja zawartości zmiennych alfanumerycznych. Pozwala to na tworzenie kilkunastu zbiorów tekstowych lub pisanie tekstów źródłowych w innych językach programowania. Część klawiszy kalkulatora zmienia w tym trybie swoje znaczenie.

W nowym modelu istotnie poprawiono własności akustyczne komputera. Układ 8912 umożliwił uzyskanie jakości dźwięku analogicznej do tej co w BBC czy Commodore 64, choć komendy BASICOWE nie zapewniają nad nim pełnej kontroli. Pozwalają za to za pośrednictwem interfejsu MIDI bezpośrednio sterować zewnętrznymi instrumentami muzycznymi, syntetyzatorami itp. Ponieważ zrezygnowano z wbudowanego głośniczka, dźwięk odtwarza się za pośrednictwem fonii telewizyjnej lub zewnętrznego wzmacniacza akustycznego.

Jak dotąd, firma nie ujawniła szczegółów stronicowania i przełączania pamięci. Prawdopodobnie nie dopracowano jeszcze ostatecznego rozwiązania. Wiadomo, że 128K pamięci podzielone będzie na strony po 16K każda, które będzie można lokować w całej przestrzeni adresowej Z 80. 32K ROMU podzielono na dwie strony. 16K zajmuje oryginalny ROM z wersji 48K a drugie 16K przeznaczono na model 128K. Niewykluczone, że konstruktorzy zdecydują się na wariant, w którym po wyborze trybu pracy odpowiedni ROM będzie kopiowany do pamięci RAM, a następnie zostanie dołączony jako pierwsza stronica. Jeśli tak, to teoretycznie możliwe będą nawet najdalej idące modyfikacje systemu, co użytkownikowi umożliwi dostęp do pełnych 128K bajtów pamięci z kodu maszynowego.

Ekran obsługiwany jest identycznie, jak w starej wersji. Obszar pamięci video zaczyna się pod adresem 16368 i niestety bez dodatkowych urządzeń zewnętrznych nie można stosować standardowego systemu CP/M.

Z poziomu BASICA dodatkowe stronice pamięci objawiają się jako tak zwane RAM-dyski. Dostęp do nich zapewniają nowe rozkazy LOAD! SAVE! W sumie użytkownik będzie miał około 104K na własne programy pisane w BASICU, choć te dłuższe będzie musiał dzielić na kawałki i ściągać z RAM-dysku w razie potrzeby. Może będzie istniała możliwość samodzielnego przełączania poszczególnych stron za pomocą instrukcji IN, OUT, PEEK i POKE.

Nie jest jasne, czy i jak nowy model będzie współpracował z urządzeniami zewnętrznymi dostosowanymi do SPECTRUM 48K. W trybie 128K mogą pojawić się problemy.

Na koniec sprawa ceny. Firma INVESTRONICA sprzedaje SPECTRUM 48K za 27 000 peset (122 funty) i SPECTRUM+ za 36 000 peset (164 funty). Przewiduje się, że SPECTRUM 128K wycenione będzie na 50 000 peset (230 funtów).

wg YOUR SPECTRUM Nr 12/85

Dzięki uprzejmości Pana Tadeusza Wilczka, autora programu "COPY-COPY", przedstawiamy instrukcję obsługi.

"COPY-COPY" jest najczęściej używanym programem dla mikrokomputera ZX Spectrum, najpopularniejszego w Polsce komputera domowego.

COPY COPY

Program kopiujący "COPY-COPY" Instrukcja obsługi

Program służy do kopiowania i przeglądania zbiorów (programów, danych i kodów – z nagłówkami lub bez) zapisanych na kasetach. Program umożliwia przekopiowanie za jednym razem do 15 zbiorów, o łącznej długości do 42496 bajtów lub jednego zbioru o długości do 49152 bajtów. Program wykonuje komendy o składni zbliżonej do języka BASIC. Słowa kluczowe uzyskuje się przez naciśnięcie jednego klawisza. Słowa wymagające zwykle przełączenia na tryb EXTENDED wywołuje się klawiszem z pierwszą literą nazwy (np. VERIFY – klawisz V). Słowo TO i STEP uzyskuje się przez SYMBOL SHIFT i F lub D. Wersja 1 programu pozwala na podłączenie ZX INTERFACE 1.

1. CAT – klawisz C
Przeglądanie nagłówków (nie niszczy zbiorów zapisanych w pamięci).

2. LOAD – klawisz J
Wczytywanie zbiorów do pamięci.
LOAD – od następnego zbioru;
LOAD 1 – od początku pamięci (skasowanie wszystkich zbiorów wczytanych poprzednio);
LOAD n – od n-tego zbioru;
LOAD n TO m – od n-tego do m-tego zbioru;
LOAD TO m – od następnego do m-tego zbioru;
LOAD AT nn – od adresu nn; nn jest inicjowany na 23296, przez LOAD AT 23040 można zwiększyć wolną pamięć do 42496; kasuje wszystkie zbiory nie niszcząc ich fizycznie;
LOAD (nn – wczytanie pierwszych nn bajtów zbioru (z pominięciem nagłówka), może być także LOAD n/nn;
LOAD (nn TO – wczytanie z opuszczeniem pierwszych nn bajtów (z pominięciem nagłówka); Przykład:
LOAD (6912 – wczytanie tylko screen-u
LOAD (6912TO – wczytanie z pominięciem screen-u

3. SAVE – klawisz S;
Nagrywanie zbiorów na taśmie.
SAVE – od pierwszego do ostatniego
SAVE n – od n-tego zbioru;
SAVE TO m – od pierwszego do m-tego zbioru;
SAVE n TO m – od n-tego do m-tego zbioru;
SAVE STEP n – ustawienie przerwy między zbiorami w sekundach, przy n = 9 (przerwa dowolna) pojawia się komunikat "Press any key ...";
SAVE n TO m STEP n – od n-tego do m-tego zbioru co n sekund;

4. VERIFY – klawisz V
Weryfikowanie nagranych zbiorów.
VERIFY – patrz LOAD;
VERIFY n TO m – patrz LOAD n TO m;
VERIFY n – patrz LOAD;

5. LET – klawisz L
Zmiana nagłówka.
LET 2 = naz,,1 – zbiór nr 2, nazwa = naz, start = 1;
LET 3 = ,500,1,5 – zbiór nr 3, dług = 500, start = 1, prog = 5;
W nazwie pomijane są spacje, można w nazwie umieścić spacje graficzne – GRAPHICS 8.
LET bez parametrów powoduje wydruk zbiorów.
Jeśli pojawia się komunikat OUT OF MEMORY, to należy zmieniać nagłówek po jednym parametrze.

6. LIST – klawisz k
Wydruk pamięci, pokazuje 15 bajtów – adres, wartość słowa, wartość bajtu /dec/ i kod znaku. Przez ENTER uzyskuje się następne 15 bajtów. LIST nn – wydruk od adresu nn.
Uwaga: zawartość obszaru pamięci pomiędzy adresami 23552 i 23754 (zmienne systemowe) jest przenoszona w inne miejsce.

7. POKE – klawisz O
Modyfikacja zawartości pamięci;
POKE x, nn x = adres, nn = wartość dwubajtowa
POKE x, n x = adres, n = wartość jednobajtowa
Jeżeli druga wartość jest większa od 256, to automatycznie wpisuje się dwa bajty.

8. USR – klawisz U
USR nn – wykonanie programu maszynowego od adresu nn.

9. RETURN – klawisz Y
Powrót do BASIC'u: inicjuje zmienne systemowe i kanały, nie niszczy fizycznie zbiorów.

10. COPY – klawisz Z
COPY – kopiowanie jednego zbioru o długości do 49096 bajtów bez nagłówka; rozpoczęcie nagrywania /SAVE/ przez CAPSSHIFT; można nagrywać wielokrotnie; pomija nagłówek; niszczy program "COPY-COPY".
COPY nn – kopiowanie jednego zbioru o długości do 49152 bajtów (bez nagłówka), początek nagrywania przez CAPSSHIFT, nagrywanie jednorazowe.
Przed wczytaniem ustawić nagranie tuż za nagłówkiem (tzn. wcisnąć ENTER dopiero po przejściu nagłówka).
COPY nn niszczy program główny "COPY-COPY", nn oznacza adres, pod którym schowany będzie program kopiujący (29 bajtów) 16384 nn 65350 np.: COPY 16387 – program schowa się w pierwszym wierszu ekranu; COPY 23698 – w MEMBOT. COPY zawiesza się w przypadku błędu na taśmie i po ukończeniu nagrywania. Moduły COPY nie są aktywne, jeśli przedtem wykorzysta się ostatnie 200 bajtów pamięci.

11. PRINT – klawisz P;
Wydruk listy zbiorów wczytanych do pamięci na drukarce SEIKOSHA GP50S lub ZX PRINTER. Aby uzyskać wydruk, należy wczytać zbiory za buforem drukarki /LOAD AT 23552/.

UWAGI:

* typy zbiorów
P – program
B – bytes
A – number array
$ – character array
■ – oznacza, że zbiór jest bez nagłówka lub że długość zbioru nie jest zgodna z długością podaną w nagłówku, na ekranie podawana jest długość rzeczywista; podczas LOAD podawana jest długość odczytywana z nagłówka, jeśli długość zgadza się, to czarny kwadrat znika.

* ENTER powtarza ostatnią komendę (bez parametrów)
* DELETE kasuje komendę
* CAPSSHIFT-7 kasuje ostatni zbiór

PĘDZEL I STOS

W trakcie programowania pozornie oczywistych, prostych czynności napotykamy często na pewną charakterystyczną trudność. Nie bardzo zdajemy sobie sprawę z tego, co komputer powinien właściwie robić. Fakt, że sami postawione zadanie rozwiązalibyśmy bez zastanowienia, stanowi raczej utrudnienie niż pomoc przy formułowaniu algorytmu. Przykładem może być często występujący w komputerowej grafice i nie tylko grafice problem zamalowywania figury ograniczonej zamkniętym konturem.

Geometria komputerowego ekranu różni się w praktyce nieco od geometrii matematycznej. Rozdzielczość ekranu jest skończona. "Punkt" ekranu posiada więc określony kształt i wymiary, a dwa punkty nie mogą znajdować się dowolnie blisko.

Przyjmijmy, że "punkt" ekranu to prostokąt o długościach boków wyznaczonych rozdzielczością ekranu (długość boków odpowiada najmniejszym odległościom między punktami w pionie i w poziomie). Najczęściej będzie to kwadrat – np. w ZX Spectrum. Za kontur zamknięty uznamy więc taki zamknięty łańcuch punktów, w którym sąsiednie punkty stykają się przynajmniej rogami (rys. 1). Zamalowanie wnętrza figury polegać będzie na wypełnieniu ("zapaleniu", zaczernieniu) wszystkich punktów zlokalizowanych wewnątrz konturu.

Zamalowanie figury z rys. 2a nie nastręczałoby problemów. Wystarczyłoby np. przebiegać, linia po linii, wszystkie punkty od lewego do prawego boku prostokąta, wypełniając je równocześnie. W przypadku figur o bardziej wymyślnym kształcie metoda ta zawiedzie. Załóżmy jeszcze, że dla zamalowania całego wnętrza dowolnej figury musi wystarczyć wskazanie dowolnego punktu należącego do wnętrza.

Ludzki zmysł wzroku ogarnia równocześnie całą figurę. Inaczej komputer. W danej chwili "widzi" on jeden, jedyny punkt ekranu i może odpowiedzieć na pytanie, czy punkt ten jest pusty (wygaszony), czy wypełniony, stanowiąc element konturu lub wcześniej zamalowanego fragmentu. Możliwość odpowiedzi na wszystkie inne pytania, dotyczące np. wzajemnego położenia kolejno analizowanych punktów, zależy od trafnego wyboru struktury danych, w jakiej zostaną one zapamiętane podczas działania programu.

Poszukiwanie algorytmu ułatwia często obserwacja procesów rozgrywających się w naszym otoczeniu. Zróbmy eksperyment: wytnijmy z papieru figurę o dowolnym kształcie i przytknijmy do niej płonącą zapałkę. Wkrótce ogień strawi cały papier, niezależnie od jego kształtu i miejsca podpalenia, pomimo że płomień przecież także nie "widzi" kartki...

Przykładając zapałkę zapalamy jeden "punkt" kartki. "Punkt" ten podpala "punkty" z nim sąsiadujące – chyba że te już też płoną. Każdy zapalony punkt zachowuje się identycznie, jak punkt podpalenia. Różnica między spalaniem papieru a zamalowywaniem figury na ekranie polega na tym, że na kartce rozprzestrzenianie płomienia w różnych kierunkach odbywa się równocześnie, zaś komputer może w danej chwili zajmować się tylko jednym punktem. Spróbujmy metodę płomienia zastosować do wypełniania obszaru (metoda znana jest pod nazwą "metody pożaru prerii").

Przyjmijmy, że każdy zapalony punkt może zapalić czterech najbliższych sąsiadów: z góry, z dołu, z lewej i z prawej. Ponieważ każdy punkt opisany jest parą współrzędnych ekranowych, wskazanie sąsiadów jest proste. Wystarczy do aktualnych współrzędnych punktu dodać lub odjąć odpowiednio 1. Po zapaleniu sąsiadów każdego z nich traktujemy podobnie jak punkt wyjściowy itd. Ponieważ jednak możemy w każdej chwili zajmować się tylko jednym z nich, współrzędne pozostałych musimy jakoś przechować, zapamiętać do czasu wykorzystania. Ponieważ na wnętrze figury składać się może wiele tysięcy, a nawet więcej punktów, sposób zapamiętywania współrzędnych musi być ekonomiczny. Pamięć powinna być zajmowana wyłącznie przez współrzędne punktów jeszcze nie obsłużonych przez komputer. Musimy też mieć możliwość stwierdzenia, że cała figura została wypełniona i można już zaprzestać pracy.

PROGRAM

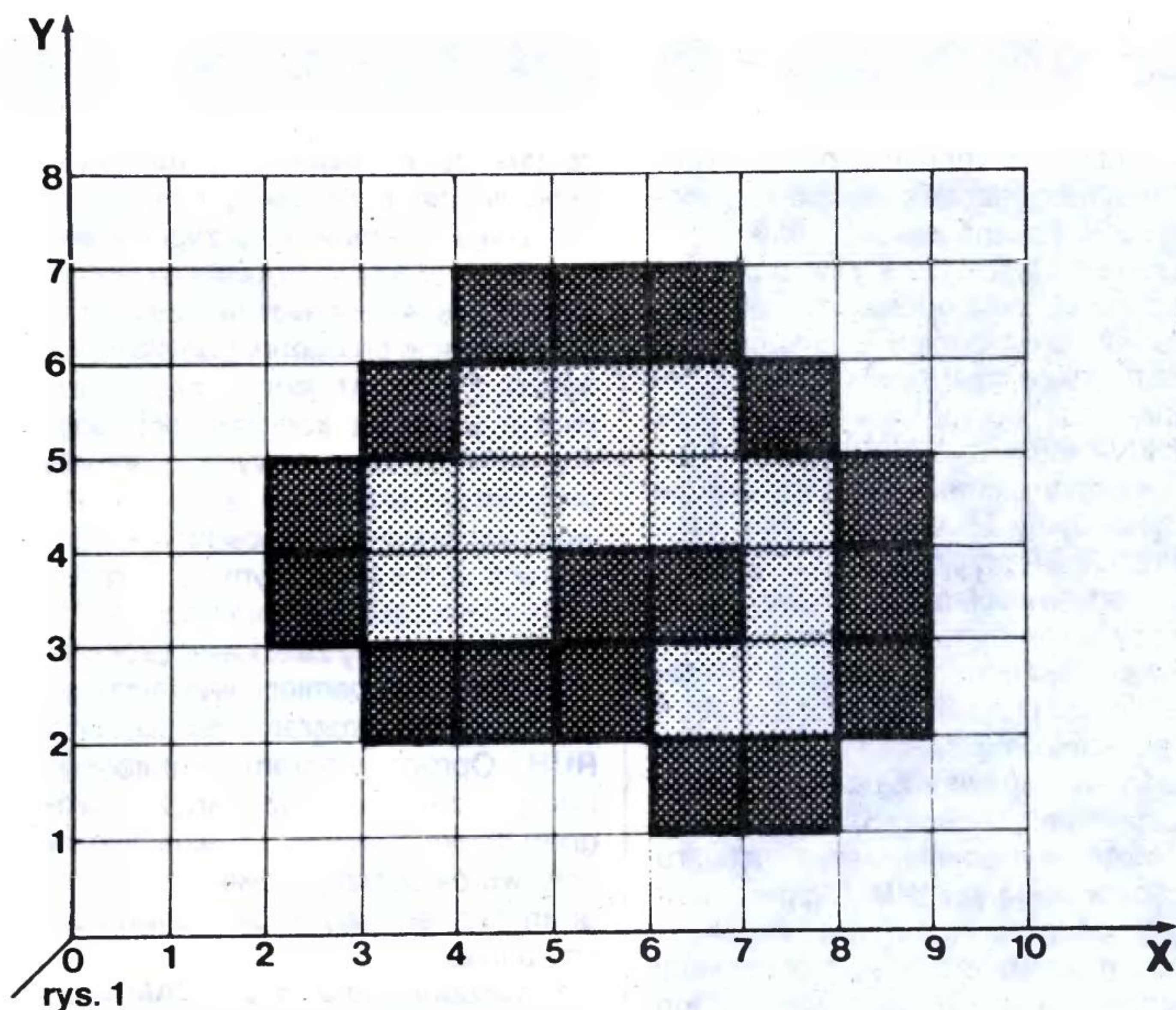
```
10 CIRCLE 128,70,20: CIRCLE 126,72,14
20 LET x=111: LET y=70: GO SUB 8000
30 PRINT "ZAMALOWANO PUNKTOW: ";P
40 STOP

7998 REM PODPROGRAM ZAMALOWYWUJACY
7999 REM Roland Waclawek      1985
8000 DIM S(200,2)
8010 LET P=0: LET SP=0
8020 GO SUB 8200
8030 IF SP=0 THEN RETURN
8040 LET X0=S(SP,1): LET Y0=S(SP,2)
8050 LET SP=SP-1
8060 LET X=X0: LET Y=Y0+1: GO SUB 8200
8070 LET Y=Y0-1: GO SUB 8200
8080 LET X=X0-1: LET Y=Y0: GO SUB 8200
8090 LET X=X0+1: GO TO 8020

8200 IF X<0 OR X>255 THEN RETURN
8210 IF Y<0 OR Y>175 THEN RETURN
8220 IF POINT (X,Y)=1 THEN RETURN
8230 PLOT X,Y: LET SP=SP+1: LET P=P+1
8240 LET S(SP,1)=X: LET S(SP,2)=Y
8250 RETURN
```

rys. 3a

```
8055 PRINT AT 0,0;SP;"   ";
```



rys. 1

Odstawmy na chwilę komputer i przymierzmy się do „ręcznej realizacji zadania". Przygotujmy plik jednakowych kartek. Na kartkach tych zapisywać będziemy współrzędne każdego zapalanego punktu: każdy punkt na oddzielnym arkuszu. Po zanotowaniu współrzędnych kartkę ułożymy na stosie podobnych kartek, zapisanych wcześniej. Jeśli zachowamy porządek, na górze stosu (na jego wierzchołku) leżeć będą zawsze współrzędne punktów zapalonych później niż punktów z kartek znajdujących się na spodzie (na dnie) stosu.

Podstawową czynnością będzie analiza otoczenia poprzednio zapalonego punktu. Badamy wszystkich czterech sąsiadów. Jeśli któryś z tych punktów jest już zapalony, to przestajemy się nim interesować. W przeciwnym razie punkt zapalamy, zaś jego współrzędne układamy na stosie.

Po przeanalizowaniu otoczenia danego punktu przystępujemy do badania otoczenia punktu następnego. Punkt ten możemy wybrać zupełnie dowolnie spośród wcześniej zapalonych, a jeszcze nie obsłużonych. Najprościej w takim razie zdjąć kartkę leżącą na szczycie stosu. Na stosie znajdują się przecież wyłącznie współrzędne punktów już zapalonych. Takie postępowanie ma praktyczną zaletę: zdjętą kartkę po wykorzystaniu możemy wyrzucić – nie będzie już potrzebna. W ten sposób nie dopuszczamy do nadmiernego wzrostu wysokości stosu. Układanie kartek na stos odbywa się na przemian z ich pobieraniem.

Zauważmy, że uzyskaliśmy też prosty wskaźnik końca pracy. Gdy na stosie nie pozostanie już ani jedna kartka, będzie to znaczyło, że nie ma już ani jednego punktu, który potencjalnie mógłby mieć nie zapalonych jeszcze sąsiadów. Pusty stos jest więc sygnałem kompletnego zamalowania wnętrza figury.

Aby zapoczątkować pracę, wystarczy jako pierwszy element stosu położyć współrzędne „punktu podpalenia"

Jeśli przystąpimy do zapisania omówionego powyżej algorytmu np. w języku BASIC – stwierdzimy, że nie ma w tym języku ani stosu, ani instrukcji odpowiadających operacjom: "POŁÓŻ NA STOSIE" i "ZDEJMIJ ZE STOSU". Tym niemniej stos możemy łatwo zaimprowizować z tablicy oraz zmiennej prostej służącej za wskaźnik

```
  10 REM *** ZAMALUJ ***  R.Waclawek
  20 CLEAR 63999: LET A=64000: LET S=0
  30 FOR I=0 TO 128: READ X: LET S=S+X
  40 POKE A+I,X: NEXT I
  50 IF S<>17271 THEN PRINT "BLAD": STOP
  60 FOR I=1 TO 4: READ X
  70 POKE A+X,A+90-256*INT ((A+90)/256)
  80 POKE A+X+1,INT ((A+90)/256): NEXT I
  90
 200 REM PRZYKLAD ZASTOSOWANIA
 210 CIRCLE 90,80,25: CIRCLE 160,80,20
 220 POKE 23728,100: POKE 23729,80
 230 RANDOMIZE USR A: PAUSE 100
 240 RANDOMIZE USR (A+9)
 250 POKE 23728,170: RANDOMIZE USR (A+3)
 260
1000 DATA 175,24,2,62,1,22,0
1010 DATA 24,3,22,255,175,217,1
1020 DATA 0,0,95,217,42,141,92
1030 DATA 34,143,92,33,192,255,237
1040 DATA 75,101,92,237,66,229,221
1050 DATA 225,6,255,197,237,75,176
1060 DATA 92,205,90,250,193,4,40
1070 DATA 35,221,229,221,57,221,225
1080 DATA 210,21,31,120,254,176,220
1090 DATA 90,250,62,254,128,71,220
1100 DATA 90,250,4,12,196,90,250
1110 DATA 62,254,129,79,56,216,24
1120 DATA 217,217,197,217,193,201,197
1130 DATA 205,170,34,60,71,172,79
1140 DATA 30,1,203,11,16,252,126
1150 DATA 170,163,121,193,192,217,3
1160 DATA 163,217,32,5,126,170,179
1170 DATA 170,119,213,205,219,11,209
1180 DATA 225,197,233,44,63,70,75
```

rys. 4

tej pozycji tablicy, w którą wpisaliśmy ostatnio nowe dane-współrzędne. Operacja "układania na stosie" polegać będzie na zwiększeniu o 1 wskaźnika stosu i zapisaniu we wskazanej komórce tablicy pary współrzędnych. "Pobieranie ze stosu" oznaczać będzie odczyt komórki wskazywanej przez wskaźnik stosu, a następnie zmniejszeniu tego wskaźnika o 1. Poprzednia zawartość odczytanej komórki pozostaje co prawda na swoim miejscu, jednak komórka ta jest zwolniona i najprawdopodobniej zostanie ponownie zapisana przy "układaniu" na stosie kolejnych współrzędnych.

Przykładowy podprogram zamalowujący przedstawia rys. 3a. Program napisano w języku BASIC dla ZX Spectrum. Zmienna **SP** to wskaźnik stosu, utworzonego z tablicy **S**. Zmienne **X0** i **Y0** przedstawiają współrzędne punktu ostatnio pobranego ze stosu, zmienne **X** i **Y** to współrzędne punktu, który aktualnie próbuje się zapalić. Zmienna **P** nie jest co prawda istotna z punktu widzenia samej operacji zamalowywania, zalicza jednak wszystkie wypełniane punkty. Po powrocie z podprogramu zmienna **P** przedstawia ilość punktów tworzących wnętrze zamalowanej figury, która jest miarą pola figury. Podprogram może więc spełnić rolę "ekranowego planimetru". Oprócz zamalowywania podprogram jest w stanie skasować z ekranu dowolną, wypełnioną figurę. Ten wariant pracy najbardziej przypomina spalanie kartki. Jedyne niezbędne modyfikacje programu to zmiana warunku (interesują nas teraz tylko punkty wypełnione) oraz zastąpienie operacji wypełniania kasowaniem punktu. Zmienna **P** jest tu także miarą pola powierzchni figury, obejmując tym razem także punkty konturu.

Pierwszy element tablicy-stosu ma indeks równy 1. Jeśli stos jest pusty, wskaźnik **SP** przyjmie więc wartość 0.

Przed wywołaniem podprogramu należy zmiennym **X** i **Y** nadać wartości współrzędnych "punktu podpalenia" (patrz przykład na rys. 3a). Przy wypełnianiu większych obszarów stos okaże się zapewne za duży i nastąpi przepełnienie tablicy **S**. Skutecznym wyjściem jest powiększenie rozmiaru tablicy w deklaracji **DIM**. Eksperymentując z podprogramem warto zapewnić sobie możliwość bieżącej obserwacji zachowania stosu, jego wzrostu lub malenia. Można to najprościej osiągnąć w drodze bieżącego wyświetlania wartości wskaźnika stosu **SP** (rys. 3b). Łatwo będzie zauważyć, że prosta algorytmu okupiona jest niestety jego dużą "stosożernością".

W większości zastosowań praktycznych działanie programu w języku BASIC okaże się zbyt wolne. Radykalnym rozwiązaniem jest zastosowanie szybkiej procedury napisanej bezpośrednio w języku maszynowym. Procedura taka na użytek naszych Czytelników została opracowana. Aby umożliwić skorzystanie z niej także tym, którzy nie znają asemblera, przygotowano specjalny program ładujący w języku BASIC (rys. 4). Pierwotnie przewidziano lokalizację procedury pod adresem 64000. Ponieważ jednak często zachodzi potrzeba koegzystencji podprogramów maszynowych z innymi programami w pamięci, istnieje możliwość umieszczenia procedury pod dowolnym adresem. W tym celu trzeba zmienić początkową wartość zmiennej **A** w linii nr 20. Aby załadować program maszynowy do pamięci, wystarczy po wprowadzeniu programu dać zlecenie **RUN**. Oprócz programu ładującego listing, zawiera też krótki program demonstracyjny. W razie błędów przy wprowadzaniu bloku DATA program ładujący wyświetli odpowiedni komunikat.

Procedura maszynowa ZAMALUJ liczy 129 bajtów. Aby po załadowaniu móc zapamiętać ją na taśmie, wystarczy zlecenie **SAVE "ZAMALUJ" CODE A,129.** Uwaga! Procedura nie jest relokowalna – nie można jej zatem wczytać do pamięci pod inny adres niż ten, spod którego została zapamiętana na taśmie. Aby przygotować wersję procedury przewidzianą do umieszczenia pod innym adresem, niezbędne będzie ponowne użycie programu ładującego z odpowiednią wartością zmiennej **A**.

```
10 ; **** ZAMALUJ ****
20 ; (C) Roland Waclawek     1985
30 ;
40 ;START1: Wypelnianie obszaru plama
50 ;START2: Wypelnianie obszaru siatka
60 ;START3: Usuwanie obszaru
70 ;

5C8D   80 ATTRP  EQU  23693
5C8F   90 ATTRT  EQU  23695
5CB0  100 WSPXY  EQU  23728
5C65  110 STKEND EQU  23653
22AA  120 PKTADR EQU  #22AA
0BDB  130 ATRYBU EQU  #0BDB
1F15  140 NORAM  EQU  #1F15
      150
FA00  160        ORG  64000
      170
FA00  180 START1 XOR  A
FA01  190        JR   START0
FA03  200 START2 LD   A,1
FA05  210 START0 LD   D,0
FA07  220        JR   START
FA09  230 START3 LD   D,#FF
FA0B  240        XOR  A
FA0C  250 START  EXX
FA0D  260        LD   BC,0
FA10  270        LD   E,A
FA11  280        EXX
FA12  290        LD   HL,(ATTRP)
FA15  300        LD   (ATTRT),HL
FA18  310        LD   HL,-#40
FA1B  320        LD   BC,(STKEND)
FA1F  330        SBC  HL,BC
FA21  340        PUSH HL
FA22  350        POP  IX
FA24  360        LD   B,#FF
FA26  370        PUSH BC
FA27  380        LD   BC,(WSPXY)
FA2B  390 NOWYP1 CALL WYPELN
FA2E  400 NOWYPU POP  BC
FA2F  410        INC  B
FA30  420        JR   Z,POWROT
FA32  430        PUSH IX
FA34  440        ADD  IX,SP
FA36  450        POP  IX
FA38  460        JP   NC,NORAM
FA3B  470        LD   A,B
FA3C  480        CP   #B0
FA3E  490        CALL C,WYPELN
FA41  500        LD   A,-2
FA43  510        ADD  A,B
FA44  520        LD   B,A
FA45  530        CALL C,WYPELN
```

rys. 5

Przed uruchomieniem procedury należy wpisać współrzędne punktu startowego do komórek RAM: X do 23728, Y do 23729. Komórki te leżą w obszarze zmiennych systemowych, przechowują jednak normalnie niewykorzystywany wektor dla przerwania niemaskowalnego (NMI). Można je spokojnie wykorzystać. Istnieją trzy możliwości wywołania procedury: od adresu **A, A+3 i A+9.** Przy wywołaniu od adresu **A**, np. ***RANDOMIZE USR A*** lub ***RANDOMIZE USR 64000.*** nastąpi wypełnianie konturu ciągłą plamą. Wywołanie od adresu **A+3**, np. ***RANDOMIZE USR 64003,*** spowoduje skasowanie wypełnionego obszaru. Procedura wywoływana jest jako funkcja. Po powrocie wartością funkcji jest ilość wypełnionych lub skasowanych punktów. W zastosowaniach "planimetrycznych" można więc podprogram wywołać przez ***PRINT USR 64000*** lub ***LET pole=USR 64003.***

Trzecie wywołanie, np. ***RANDOMIZE USR 64009,*** także powoduje zamalowywanie wnętrza konturu. Tym razem jednak wnętrze figury zamalowywane jest nie plamą, lecz siatką. Wartość funkcji nie odpowiada tu już polu figury, a zapotrzebowanie na pamięć roboczą jest znacznie większe.



rys. 2

Procedura maszynowa ZAMALUJ korzysta z tzw. stosu systemowego. Procesor Z 80, podobnie jak większość innych mikroprocesorów, dysponuje integralnym wbudowanym wskaźnikiem stosu, co upraszcza sprawę. W odróżnieniu od przykładu w BASIC-u, stos w Z 80 przyrasta jednak nie w górę, lecz w dół pamięci. W razie przepełnienia stosu systemowego nie grozi zniszczenie programu: zostanie wyświetlony komunikat o błędzie **"Out of memory"**. Nastąpi to jednak tylko wtedy, gdy stos będzie zlokalizowany poniżej procedury ZAMALUJ. Dlatego też przed załadowaniem programu należy bezwzględnie wykonać instrukcję CLEAR z argumentem przynajmniej o 1 mniejszym niż początkowy adres ładowania **A** (patrz rys. 4).

Dla Czytelników wprawionych w programowaniu w języku asemblera Z 80 przytaczamy tekst źródłowy programu ZAMALUJ dla popularnego asemblera GENS 3 (rys. 5). Na początku na stos ładowana jest jako wartość współrzędnej Y stała 255. W praktyce taka wartość wystąpić nie może – pobranie ze stosu Y=255 oznacza więc opróżnienie stosu. Bieżąca wartość współrzędnej X przechowywana jest w rejestrze C, współrzędnej Y – w B. Para **BC'** służy za licznik wypełnionych lub zgaszonych punktów.

Roland Wacławek

```
FA48 540          INC  B
FA49 550          INC  C
FA4A 560          CALL NZ,WYPELN
FA4D 570          LD   A,-2
FA4F 580          ADD  A,C
FA50 590          LD   C,A
FA51 600          JR   C,NOWYP1
FA53 610          JR   NOWYPU
     620
FA55 630 POWROT   EXX
FA56 640          PUSH BC
FA57 650          EXX
FA58 660          POP  BC
FA59 670          RET
     680
FA5A 690 WYPELN   PUSH BC
FA5B 700          CALL PKTADR
FA5E 710          INC  A
FA5F 720          LD   B,A
FA60 730          XOR  H
FA61 740          LD   C,A
FA62 750          LD   E,1
FA64 760 WYSUW    RRC  E
FA66 770          DJNZ WYSUW
FA68 780          LD   A,(HL)
FA69 790          XOR  D
FA6A 800          AND  E
FA6B 810          LD   A,C
FA6C 820          POP  BC
FA6D 830          RET  NZ
FA6E 840          EXX
FA6F 850          INC  BC
FA70 860          AND  E
FA71 870          EXX
FA72 880          JR   NZ,KONIEC
FA74 890          LD   A,(HL)
FA75 900          XOR  D
FA76 910          OR   E
FA77 920          XOR  D
FA78 930          LD   (HL),A
FA79 940 KONIEC   PUSH DE
FA7A 950 MODYF    CALL ATRYBU
FA7D 960          POP  DE
FA7E 970          POP  HL
FA7F 980          PUSH BC
FA80 990          JP   (HL)
```

WIEŻA BABEL,

CZYLI JĘZYKI PROGRAMOWANIA

Wyobraźmy sobie młodego człowieka, szczęśliwego posiadacza świeżo nabytego mikrokomputera. Dotąd z informatyką stykał się jedynie pośrednio. Oglądał programy telewizyjne, czytał kąciki mikroinformatyczne w prasie, a nawet udało mu się skompletować wszystkie numery Bajtka. Przez cały ten czas marzył o własnym sprzęcie i wreszcie się udało. Ktoś rozsądny ostrzegł go kiedyś, by nie próbował uczyć się żadnego języka programowania na sucho czyli bez dostępu do maszyny. Wtedy usłuchał, ale teraz ma już to swoje ukochane cacko i chciałby zacząć programować.

Z telewizji pamięta, że świetnym językiem jest LOGO. Kolega ma PASCAL i C, a koleżanka FORTH (wymowa: [for :0]). Bajtek opisał również PROLOG, w komputerze ma BASIC, a w głowie kompletny zamęt. I dobrze, że nie wie, iż na świecie używa się co najmniej 300 dalszych różnych języków, bo by chyba zupełnie zwariował.

Tak jak dla posiadaczy komputerów osobistych najlepszym modelem jest zazwyczaj ten właśnie posiadany, podobnie dla programujących najlepszym językiem jest ten, który dobrze znają. Z jednym może wyjątkiem – prawie wszyscy zgodnie krytykują BASIC (wymowa: ['beisik]).

Zadziwiające, jak niewielu ludzi, szczególnie wśród popularyzatorów różnych języków, dostrzega lub chce dostrzegać ich rolę jako narzędzia i to jednego z wielu. A przecież pytanie, który język jest lepszy często ma tyle samo sensu, co debata o wyższości młotka nad obcęgami. Odpowiedź oczywiście zależy od tego, do jakich celów dane narzędzie ma być wykorzystane. Nikt rozsądny nie będzie używał mikroskopu do wbijania gwoździ. Również rozsądny programista nie użyje LISP do przybliżonego rozwiązywania równań różniczkowych. A przecież teoretycznie oba zastosowania są możliwe.

NA POCZĄTKU BYŁ CHAOS

Kiedyś z językami programowania nie było problemów, jako że nie było żadnych języków. Pierwsze komputery programowało się w czystym kodzie maszynowym wprowadzając do niego pracowicie nieskończone ciągi zer i jedynek. Sztuka programowania dostępna była jedynie bardzo wąskiemu gronu wtajemniczonych. Wydajność programistów, jak na dzisiejsze standardy, była żenująco niska. Proces uruchamiania i usuwania błędów z programów był prawdziwą katorgą możliwą do zniesienia jedynie dla prawdziwych zapaleńców. Do tego kody maszynowe dla różnych komputerów były różne, zależne od konkretnej konstrukcji i o przenoszeniu programów między nimi nie mogło nawet być mowy.

Wzrost zapotrzebowania na usługi komputerowe wymusił szybki postęp. Najpierw powstały asemblery, zwane niekiedy autokodami. Pojedyncze instrukcje zaczęto zapisywać nie w postaci ciągów zer i jedynek, a jako skrócone nazwy rozkazów. Liczby binarne (dwójkowe) zastąpiono heksadecymalnymi (szesnastkowymi) i napisano programy tłumaczące to wszystko z powrotem na kod maszynowy. Dodano kilka ułatwień dla programistów, umożliwiając im posługiwanie się adresami symbolicznymi i tak zwanymi makro-instrukcjami, czyli możliwością oznaczania jedną nazwą całego ciągu instrukcji. Niby niewiele, a jednak dało to narzędzie tak dobre, że do dzisiaj jest stosowane wszędzie tam, gdzie jest konieczne panowanie nad całym komputerem, przy oprogramowaniu współpracy maszyny z urządzeniami zewnętrznymi, jak również w tych zastosowaniach, w których bardzo istotna jest efektywność programu, a więc jego szybkość i oszczędność wykorzystania pamięci. Wadą asemblerów pozostało ich przywiązanie do konkretnego modelu procesora. Użycie ich wymaga na ogół znajomości budowy danego typu komputera oraz sporego doświadczenia w programowaniu. Możliwości popełniania fatalnych błędów są tu praktycznie nieograniczone, a czytelność programów jest odwrotnie proporcjonalna do ich długości. Na pewno nie są to języki dla początkujących, a i zaawansowanym programistom bywają obecnie potrzebne jedynie w szczególnych sytuacjach.

CZAS TO PIENIĄDZ

Dalszy rozwój języków programowania był równie szybki, co samego sprzętu. Spadek kosztów produkcji komputerów doprowadził do sytuacji, w której wartość oprogramowania często wielokrotnie przekracza cenę samej maszyny. Zaczął nabierać znaczenia czas przygotowywania i uruchamiania programów. To z kolei wiązało się z koniecznością zapewnienia programistom maksymalnej wygody pracy. Wzrastająca liczba komputerów i osób z nich korzystających lub chcących korzystać postawiły przed twórcami języków programowania wysokie wymagania. Uważa się, że dobry język to ten, który człowiek opanuje z łatwością, niezależnie od konkretnego komputera, dający możliwość strukturalnego pisania programów, ułatwiający korzystanie z bibliotek podprogramów, umożliwiający pisanie efektywnych kompilatorów dających zwięzły i szybki kod wynikowy, auto-

matycznie wykrywający błędy programisty i charakteryzujący się czytelnością tekstów programów. W językach ogólnego przeznaczenia dodatkowo rozpatruje się ich faktyczną uniwersalność, a więc łatwość programowania skomplikowanych operacji na bardzo złożonych strukturach danych.

Na korzyść fanatycznych zwolenników poszczególnych języków działa okoliczność, że ideału nie ma. Co więcej, niemal każdy język występuje często w wielu odmianach, niejednokrotnie znacznie się od siebie różniących. Szczególnie bogaty w rozmaite dialekty jest BASIC; praktycznie co firma – to inny język! Utrudnia to znacznie praktyczną ocenę danego języka, gdyż w dyskusji trzeba się stale zastrzegać, o jaką wersję chodzi.

Jak zwykle w sporach teoretycznych, najlepszym weryfikatorem jest samo życie. Pomińmy zatem języki tworzone do specjalnych zastosowań, koncentrując się na tych, które są w miarę uniwersalne i już zdobyły sobie zwolenników wśród szerokiego grona użytkowników mikrokomputerów.

Zanim dokonamy takiego przeglądu, trzeba sobie uświadomić, w jaki sposób programy są wykonywane w maszynie. W zasadzie są stosowane dwie metody. W pierwszej program napisany w danym języku jest tłumaczony za pomocą specjalnego programu zwanego kompilatorem lub translatorem na język wewnętrzny danego komputera. Powstaje wtedy tak zwany kod wynikowy i dopiero on wykonuje zadania postawione przez programistę. Druga metoda – to stosowanie programów zwanych interpreterami. Czytają one kolejne instrukcje programu źródłowego i jeśli są wolne od błędów – wykonują je. Czasem też stosuje się technikę mieszaną. Oba sposoby mają swoje zalety i wady, rzutujące głównie na efektywność programów. Kompilatory zapewniają zazwyczaj znacznie krótszy czas wykonania i pozostawiają tłumaczonemu programowi więcej wolnej pamięci. Za to interpretery znacznie ułatwiają pisanie, uruchamianie i testowanie programów. Umożliwiają również pracę z komputerem w trybie konwersacyjnym. Informacja, w jakim trybie dany język jest realizowany, pozwala we właściwym świetle widzieć pewne jego cechy i możliwości. Warto jednak zdawać sobie sprawę, że wraz ze wzrostem mocy obliczeniowej współczesnych mikrokomputerów, a więc szybkości działania i pojemności dostępnych pamięci, problemy te będą traciły stopniowo znaczenie.

Zacznijmy od zamierzchłej historii, a więc języków takich jak FORTRAN, COBOL, ALGOL, PL/I, LISP. Powstały one przed lub w okolicach 1960 roku w erze dużych komputerów, gdy przyszłego istnienia dzisiejszych "maluchów" nikt jeszcze nie podejrzewał. Pierwsze cztery z nich były niewątpliwie najpopularniejsze i do dzisiaj są jeszcze z powodzeniem stosowane.

FORTRAN i COBOL

Zawdzięczają to nie tyle swym zaletom, co faktowi, że przez lata ich królowania stworzono ogromne biblioteki oprogramowania, jakimi konkurencyjne języki nie mogą się jeszcze pochwalić. FORTRAN jest językiem przeznaczonym do obliczeń numerycznych, COBOL zaś jest stosowany do przetwarzania dużych ilości danych wymagających jedynie prostych operacji rachunkowych. Żaden z nich nie nadaje się do przejęcia funkcji drugiego.

ALGOL

Pewnym krokiem naprzód było opracowanie ALGOLU. Jest to również język nastawiony na obliczenia numeryczne, ale jego możliwości są większe od FORTRANU. Przez pewien czas ALGOL był językiem publikacyjnym, zanim nie został wyparty przez PASCAL. W praktyce nie zdążył się tak rozpowszechnić, jak jego starsi bracia FORTRAN i COBOL. Pojawianie się ich na mikrokomputerach należy traktować raczej jako ciekawostki, bo lata świetności mają już poza sobą.

PL/I

PL/I opracowano w firmie IBM. Uwzględniono w nim doświadczenia użytkowników języków FORTRAN i COBOL, co pozwoliło stworzyć język doskonalszy i bardziej uniwersalny od poprzedników. Znaczenie samej firmy i fakt, że wszystkie swoje komputery wyposażała w kompilatory PL/I oraz programy wspomagające i ułatwiające pracę programistów, przyczyniły się znacznie do jego spopularyzowania. A nie jest to wcale zły język. Pozwala na strukturalizację programów, działania na tekstach są w nim dosyć łatwe do przeprowadzania; obsługę kanałów wejścia/wyjścia oraz współpracę ze zbiorami zewnętrznymi ma rozwiązane lepiej niż PASCAL. Ważną zaletą języka jest jego duża przenośność między różnymi komputerami. Pracuje równie dobrze w systemie CP/M na ośmiobitowych mikrokomputerach, jak i na potężnym IBM 370. Mimo tych zalet wielu użytkowników uważa PL/I za język trudny do opanowania, a programy w nim pisane za mało czytelne. Są to oczywiście odczucia względne. Rozbudowany zestaw instrukcji (większy niż w PASCALU) jest dla zwolenników zaletą, a dla przeciwników wadą. W miarę zacierania się różnic między mini- i mikrokomputerami popularność tego języka może wzrastać.

W odróżnieniu od dotychczas omówionych języków, dwa następne przeznaczone są do pracy konwersacyjnej.

APL

APL zorientowany jest głównie na obliczenia numeryczne. Ma wbudowaną dużą liczbę rozmaitych funkcji i operacji matematycznych i logicznych, co predystynuje go do roli potężnego kalkulatora. Jest to jednak jego jedyna zaleta. Alfabet APL zawiera całą masę nietypowych symboli, jak kółeczka, trójkąciki, gwiazdki itp. co istotnie utrudnia jego naukę i zapamiętanie. Wśród wielu użytkowników APL ma opinię języka, w którym można programy tylko pisać: w godzinę po napisaniu teksty stają się absolutnie nieczytelne nawet dla autorów! Za to w jednej linii można pomnożyć dwie macierze, odwrócić wynik i jeszcze zostanie miejsce na kilka dalszych operacji. Na powodzenie APL może jedynie liczyć wśród tych, którym potrzebne jest potężne liczydło do doraźnych skomplikowanych rachunków.

LISP

LISP dla odmiany jest językiem przeznaczonym do nietypowych obliczeń nienumerycznych. Największe uznanie zdobył sobie w ośrodkach pracujących nad sztuczną inteligencją. Jest on świetnym narzędziem do przetwarzania tak zwanych struktur listowych. Programy pisane w LISP doskonale sobie radzą z obliczeniem różniczek czy całek nieoznaczonych i to nie przybliżonym a dokładnym, na wzorach! Automatyczne dowodzenie twierdzeń, wnioskowanie logiczne to również dziedziny, w których króluje LISP. Ponieważ programy pisane w tym języku mogą same siebie modyfikować w czasie wykonywania, wymusza to niejako ich pracę za pośrednictwem interpreterów. Ponadto LISP rozpycha się w tak wielkich przestrzeniach pamięci, że przynajmniej na małych mikrokomputerach do poważnych zastosowań się nie nadaje. Obecne wersje tego języka są trudne do nauczenia się, teksty programów dla niewprawnych czytelników są bardzo mało czytelne (straszliwe liczby nawiasów!). Świetność LISP i upowszechnienie się go wśród szerokiego kręgu użytkowników jeszcze nie nadeszła. Być może, zanim szybkość i pojemność mikrokomputerów dorosną do potrzeb tego języka, powstaną już jego następcy. LISP bowiem stale się jeszcze rozwija, a wraz z tym rośnie zakres jego zastosowań.

BASIC

Od obliczeń symbolicznych i sztucznej inteligencji wróćmy na ziemię, do języka BASIC. Nazwa pochodzi od angielskich słów „Beginer's All Purposes Symbolic Instruction Code" (wymowa: bi'gines o:l 'pe:peses sim'bolik in'struksen koud). W swobodnym tłumaczeniu oznacza to uniwersalny język programowania dla początkujących. Przeznaczony jest do rozwiązywania szerokiej gamy problemów naukowo-technicznych. Choć powstał w latach sześćdziesiątych, dotąd nie uzgodniono jego standardowej wersji.

Jest to język niewątpliwie najłatwiejszy do szybkiego opanowania i tylko temu zawdzięcza swą największą popularność. Niemal wszystkie maszyny są wyposażone w interpreter jakiegoś dialektu języka BASIC. Opracowano cały szereg jego kompilatorów. Trzeba znać wersję języka BASIC z własnego komputera, ale programować w nim poważnych rzeczy raczej nie należy (chyba że dysponuje się dobrą jego wersją jak np. True BASIC i ograniczenia języka nie wymuszają stosowania specjalnych trików). Świetnie się natomiast nada do przeprowadzania szybkich, niewielkich podręcznych obliczeń. W miarę wzrostu trudności problemu i długości programu zaczną się objawiać jego wady. Najważniejsza z nich to ograniczone liczby dopuszczalnych typów danych. Wiele wersji narzuca ograniczenia na liczbę i sposób nazywania zmiennych. Również możliwość strukturalizacji programów jest w wielu przypadkach tylko pozorna, ze względu na brak możliwości lokalizacji zmiennych w podprogramach. Wiele dialektów wręcz prowokuje do bardzo niechlujnego stylu programowania. Czytelność programów naszpikowanych instrukcjami GO TO wymowa: (gou tu:) jest zawsze co najmniej problematyczna. Często gdy zachodzi konieczność modyfikacji programu okazuje się, że mniej pracochłonne jest napisanie go od nowa. Choć stale pojawiają się coraz nowsze i doskonalsze wersje języka BASIC, pozostanie on językiem dla początkujących i tych, którzy w zasadzie nie muszą ani nie chcą programować niczego, poza być może podręcznym kalkulatorem. Stale rosnąca liczba specjalistycznych programów pisanych przez profesjonalistów już doprowadziła do sytuacji, że na ogół prościej i taniej jest kupić gotowy program realizujący zadane cele niż pisać go samemu. W przypadku programów w języku BASIC ze względu na mnogość dialektów są małe szanse na zwrócenie sobie poniesionych kosztów przez rozpowszechnienie własnych dzieł. Na ogół zresztą BASIC nie pozwala na uzyskanie jakości efektu końcowego porównywalnej z programami pisanymi w innych językach.

PASCAL

W znacznej mierze wolny od tych wad jest drugi co do popularności język, PASCAL. Opracowany przez Niklausa Wirtha w 1973 roku z miejsca zdobył sobie niezwykłe powodzenie. Jest to język bardzo zwięzły i prosty, a jednocześnie pozwala łatwo programować bardzo skomplikowane operacje na niezwykle złożonych strukturach danych. Niemal natychmiast opanował wyższe uczelnie, gdzie jest wykładany na pierwszym roku jako podstawowy język programowania. Przejął też rolę języka publikacyjnego, co oznacza, że większość algorytmów publikowanych w książkach i pismach specjalistycznych zapisywana jest właśnie w PASCAL. Programy pisane w tym języku są niemal samodokumentujące się. Minimalna liczba komentarzy w tekście wystarcza, by był on dobrze czytelny dla innych. Bez przesady można stwierdzić, że PASCAL stał się wzorem elegancji, prostoty i uniwersalności. Wiele języków opracowywanych później (zarówno specjalizowanych jak i ogólnego przeznaczenia) w mniejszym lub większym stopniu wzorowało się na języku PASCAL i jego konstrukcjach. W odróżnieniu od języka BASIC język ten znacznie lepiej nadaje się do pisania programów użytkowych. Jego struktura wymusza porządny, logiczny, strukturalny styl programowania. W wielu przypadkach mechanizmy wbudowane w język pozwalają jeszcze na etapie kompilacji wykryć większość błędów programisty.

Dla równowagi należy również wspomnieć o jego wadach. W zasadzie jest to język przeznaczony do pracy z kompilatorem. Uruchamianie programów jest zatem bardziej kłopotliwe i pracochłonne niż przy pracy z językami interpretowanymi. Są kłopoty z tworzeniem bibliotek skompilowanych programów. Na ogół trzeba dołączać do tekstu programy źródłowe i po niewielkich modyfikacjach kompilować całość. Wersja standardowa języka bardzo skąpo traktuje opis procedur wejścia/wyjścia oraz sprawę współpracy programu ze zbiorami zewnętrznymi, pozostawiając te "detale" konstruktorom konkretnych realizacji. Ich pomysłowość zaś doprowadziła do szybkiego rozmnożenia się różnych rozszerzeń języka PASCAL, przez co zmniejszyła się łatwość przenoszenia gotowych programów między różnymi komputerami. Na szczęście nie wystąpiło to w takiej skali, co w przypadku języka BASIC.

MODULA-2

Jak już wspominaliśmy wcześniej, nie ma języka doskonałego, zdolnego zadowolić wszystkich. Ulepszoną wersją języka PASCAL jest nowe opracowanie Niklausa Wirtha – MODULA-2. W Polsce jest to jeszcze język egzotyczny, ale wszystko wskazuje na to, że już wkrótce MODULA-2 może zająć miejsce języka PASCAL. Zwłaszcza że programy pascalowskie można bardzo łatwo przerobić na nowy język i przeszło 90 % pracy może być automatycznie wykonane przez komputer.

PROLOG

Kolejnym językiem godnym uwagi jest PROLOG. Najkrócej różnicę między nim a innymi językami można ująć w ten sposób: pisząc program np. w PASCAL instruuje się komputer jak działając krok po kroku ma rozwiązać nasz problem, w PROLOGU zaś definiuje się, co ma rozwiązać i po podaniu niezbędnych przesłanek i danych, pyta się o odpowiedź. Jest to język programowania logicznego. Współczesne jego realizacje są jeszcze dalekie od doskonałości i przez to liczba jego zastosowań jest jeszcze niewielka. Efektywność programów pisanych w PROLOG-u nie wytrzymuje jeszcze konfrontacji z wymaganiami. Na razie stosuje się go do oprogramowywania robotów, budowy systemów do porozumiewania się człowieka z maszyną przy pomocy języka naturalnego, do konstruowania niewielkich relacyjnych baz danych. O tym jak fachowcy widzą przyszłość PROLOG-u może świadczyć fakt, że Japończycy prowadząc prace badawcze nad komputerami V generacji już teraz zapowiadają, że właśnie PROLOG będzie ich głównym językiem systemowym, a więc takim jak asembler dla współczesnych maszyn.

C

Na dzisiaj rolę takiego języka systemowego pełni, obok asemblerów, język C (wymowa: si). Rozpowszechnił się wraz z systemem operacyjnym UNIX (wymowa: ju:niks). Stworzony został właśnie na potrzeby ludzi zajmujących się budową systemów operacyjnych. Niewątpliwej popularności C przysparza również fakt, że będąc językiem wyższego rzędu pozostawia programiście niemal tyle samo swobody i możliwości co asembler, a przy tym jest to chyba najdoskonalszy język, jeśli rozpatrywać jego przenośność. Programy pisane w C niemal bez żadnych modyfikacji będą równie dobrze działać na ZX SPECTRUM, jak i na największym i najszybszym komputerze świata – Cray. Odznaczają się one wyjątkową zwięzłością, przez co dla początkujących mogą być mało czytelne. C zapewnia możliwość strukturalizacji programu, pozwala pracować praktycznie z dowolnymi strukturami danych. Łączy w sobie możliwości asemblera i PASCALA. Jest jednak językiem bardzo wyraźnie przeznaczonym dla fachowców. W odróżnieniu od PASCALA kompilator C z reguły zakłada, że człowiek zawsze ma rację i podejmuje się wykonania operacji czasem, delikatnie mówiąc, dziwnych. Z jednej strony stwarza to dodatkowe możliwości programistom, z drugiej jednak pozwala na popełnianie niemal dowolnych błędów, czasem bardzo subtelnych i trudnych do wykrycia, co mniej doświadczonych użytkowników może czasem wpędzić w głęboką frustrację. Opanowanie go nie jest tak proste jak PASCALA czy BASICA. Często kompilatory pracujące poza otoczeniem systemu operacyjnego UNIX mają tendencję do produkowania nadmiernie długich kodów wyniko-

wych, co może w praktyce ograniczać możliwości tego języka na najmniejszych mikrokomputerach. Jest to jeszcze język młody, rozwijający się. Grono jego zwolenników rośnie, poprawia się również jakość dostępnych kompilatorów. Jego rola jako głównego narzędzia fachowców do pisania najrozmaitszych programów użytkowych długo jeszcze będzie nie zagrożona.

FORTH

Do języków narzędziowych można również zaliczyć FORTH. Opracowano go z myślą o użytkownikach mikrokomputerów. Przez pewien czas był głównym językiem programowania pracowników firmy Atari. Pracuje się z nim w trybie interakcyjnym, mogąc jednak na bieżąco kompilować opracowane i sprawdzone fragmenty. Programowanie w FORTH polega na kolejnym definiowaniu słów, czyli własnych komend, uzupełniając w ten sposób zestaw podstawowy. Definicje te po skompilowaniu są traktowane przez system jak własne. Pozwala to tworzyć wersje języka przystosowane do indywidualnych potrzeb. Ponadto ważną zaletą tego języka jest produkowanie bardzo oszczędnego i szybkiego kodu wynikowego. Podobnie jak C, zapewnia szereg możliwości zazwyczaj zastrzeżonych dla asemblera i niedostępnych w innych językach. Jest więc stosowany do pisania programów sterujących urządzeniami zewnętrznymi, maszynami itp. Również wiele gier video na automatach napisano przy pomocy tego języka. Popularność FORTH ogranicza się do grona profesjonalnych programistów. Nauczenie się go jest sprawą trudną. Mało, że korzysta z tak zwanej n o t a c j i   p o l s k i e j   w y r a ż e ń   (patrz obok), czyli nie stosuje się nawiasów, to wszystkie zmienne i wyniki pośrednie przechowuje na stosie pozostawiając programiście problem panowania nad nim. Wymaga od piszącego wyjątkowo dużej dyscypliny. Ponadto zawiera rozbudowany zestaw instrukcji, który trudno jest zapamiętać i bardzo łatwo zapomnieć.

LOGO

Na koniec pozostawiliśmy sprawę LOGO. Wokół tego języka narosło wiele mitów i nieporozumień. LOGO należy traktować jako język specjalizowany. Powstał po to, by ułatwić (głównie dzieciom) oswojenie się z komputerem i naukę podstaw informatyki; spełnia te funkcje wyśmienicie. Pracuje się z nim za pośrednictwem interpretera, co pozwala natychmiast obserwować wyniki własnych działań i korygować ewentualne błędy. Daje dużą swobodę we wprowadzaniu do systemu słownictwa naturalnego, przez co jest łatwy do opanowania, programy w nim pisane odznaczają się dużą przejrzystością i czytelnością. Automatycznie wdraża dobre nawyki programowania strukturalnego. Ma charakter języka logicznego bardziej zbliżonego do PROLOGU czy LISPU niż PASCALA. Wbudowane mechanizmy pozwalają łatwo demonstrować różnorodne zastosowania komputerów, choć na ogół na niezbyt skomplikowanych czy rozbudowanych przykładach. Posiada również dobrze rozwiązaną grafikę. Nie jest to jednak język graficzny. Na sterowanie żółwiem przeznaczona jest tylko niewielka część jego zestawu podstawowych instrukcji. Jest to po prostu wspaniały język edukacyjny do wspomagania nauki podstaw informatyki, ale absolutnie nie nadaje się do praktycznego rozwiązywania skomplikowanych problemów, a zwłaszcza numerycznych czy opartych na przetwarzaniu dużych ilości danych. Może w przyszłości, gdy rozpowszechnią się szybsze komputery o większych pamięciach, wzrośnie rola jakiejś bardziej rozbudowanej wersji LOGO, ale na to jeszcze musimy poczekać.

Jak widać, jest z czego wybierać w wieży Babel. Przypomnijmy jednak, że na ogół co komputer, to inna realizacja danego języka. Na przykład na ZX SPECTRUM zarówno FORTH jak i C są dostępne jedynie w wersjach całkowitoliczbowych, zaś PASCAL w swojej odmianie na pewno nie jest lepszy od oficjalnego standardu. Ponadto chcąc studiować jakiś język, trzeba mieć dostęp do źródeł, najlepiej w języku polskim. Z tym zaś na naszym rynku jest fatalnie. Instrukcje fabryczne kompilatorów na ogół odsyłają do powszechnie dostępnych za granicą podręczników. O ile do FORTRANU, COBOLU, PL/I, LISPU, PASCALA i PROLOGU istnieją przynajmniej pojedyncze podręczniki, to już z FORTHEM, C, MODULA-2, APL, jesteśmy całkowicie zdani na podręczniki obcojęzyczne – jeśli potrafimy dotrzeć do biblioteki, która je posiada. Czasem można trafić na wewnętrzne wydawnictwa ośrodków obliczeniowych zawierające opisy różnych wersji interesujących nas języków. Pojawiające się czasem na perskim targu kserograficzne odbitki tłumaczeń książek zachodnich czasem przerażają niską jakością i zawsze ceną!

Niestety, krajowi wydawcy ciągle jeszcze traktują entuzjastów mikroinformatyki po macoszemu i nie dostrzegają ich potrzeb. Pozostaje nadzieja, że wprowadzenie informatyki do szkół oraz wzrost ilości sprzętu w kraju pobudzi wydawnictwa do śmielszego wkroczenia na tę pustynię.

Andrzej Kadlof

NOTACJA POLSKA

Notacja polska, a dokładnie notacja polska odwrotna, to system zapisu wyrażeń arytmetycznych, pozwalający zapisać dowolnie skomplikowane wyrażenie bez korzystania z nawiasów lub innych znaków zmieniających ustaloną kolejność działań. W skrócie polega ona na podaniu najpierw argumentów danego działania lub operacji, a potem dopiero symbolu samej operacji, przy czym argument może mieć postać liczby podanej wprost lub być również opisany w postaci wyrażenia arytmetycznego. Np. wyrażenie 5 + 7/12 zapisane w notacji polskiej ma postać 5 7 + 12 /, czyli pięć i siedem dodać, a wynik tej operacji i dwanaście podzielić.

Zapis ten opracował logik i filozof, jeden z twórców lwowsko-warszawskiej szkoły matematycznej, Jan Łukasiewicz (1878-1956). Był on m.in. twórcą logiki trójwartościowej. Z początku nowa notacja uchodziła jedynie za ciekawostkę, jej światowa kariera zaczęła się wraz z rozwojem informatyki. Okazała się ona bowiem wyjątkowo dobrze dostosowana do wymogów komputerów: maszyna mogła przystępować do wykonywania odpowiedniej operacji natychmiast po rozpoznaniu jej symbolu, gdyż argumenty znała już uprzednio. W dodatku sposób "nawarstwiania się" argumentów, z których zawsze tylko dwa ostatnie brane są pod uwagę podczas wykonywania kolejnej operacji, był w pełni zgodny z organizacją pamięci maszyny w tzw. stos czyli strukturę działającą jak stos kartek na stole. Jako pierwszy zdejmowany jest zeń przedmiot położony nań jako ostatni (Anglicy nazywają to strukturą LIFO: Last In First Out). Maszyna dysponująca stosem może więc obliczać zapisane w tej notacji wyrażenia bez angażowania choćby jednej komórki pamięci, bez straty choćby jednego kroku na organizowanie i pamiętanie informacji o konstrukcji wyrażenia: nawiasach, kolejności działań itp.

Kalkulatory firmy Hewlett-Packard będące przed 10 laty przebojem w taki właśnie sposób wykonywały rachunki. Ci, których zmusiły one do opanowania nowego zapisu i wręcz myślenia w ten sposób o wyrażeniach arytmetycznych – wiedzą, jak wspaniale jest on przejrzysty i naturalny w budowie.

Z czasem pojedyncze komórki pamięci i cykle zegara przestały być tak nadzwyczaj cenne. Komputery przystosowały się więc do starych nawyków ludzi i nauczyły się liczyć po naszemu. Notacja polska, tym razem nie odwrotna, a prosta, dokładnie w postaci wymyślonej przez Łukasiewicza, nie została zapomniana. W językach listowych, w tym w LOGO, jest ona podstawą już nie tylko obliczania wyrażeń arytmetycznych, lecz całej składni języka. Nie posługuje się on bowiem, jak wiadomo, żadnymi znakami przestankowymi. Interpreter LOGO wie, że po wywołaniu pewnych procedur powinny być podane ich parametry: jeden, dwa, trzy lub więcej. Parametry mogą być podane wprost lub jako wynik działania kolejnej procedury, której parametry znów mogą być wynikiem działania innych procedur.

Tak myśl polskich matematyków, którzy jak się wydawało współczesnym, spekulowali w jak najdalej odległych od praktyki dnia codziennego obszarach, leży dziś u podstaw języka, od którego setki milionów ludzi na całym świecie, rozpoczynają poznawanie nowego sposobu komunikowania się, nowego pisma naszej cywilizacji-informatyki.

(W. Maj)

Steve Wozniak po opuszczeniu na początku ub. r. firmy APPLE COMPUTER założył własną firmę CL9, lecz nie było wiadomo, czym się będzie zajmować. Kurtyna poszła w górę w pierwszej połowie stycznia br., gdy na targach elektroniki domowej Wozniak zaprezentował uniwersalne urządzenie sterujące wszystkim co gra i pokazuje obraz w domu zamożnego Amerykanina.

W USA wszystko musi być zdalnie sterowane. Taka panuje moda. Użytkownik wszakże staje przed wyborem: albo kupi jedno urządzenie z dziesiątkami malutkich guziczków, albo liczne urządzenia będą zagracać mu stolik do kawy...

Zdalne sterowanie (bez kabla, oczywiście) zostało wprowadzone na rynek przez firmę Zenith w 1955 r. Najpierw wykorzystywało ultradźwięki. Wygoda duża – nie trzeba wstawać z fotela by zmienić kanał w TV, ale dzwonek telefonu czy brzęk łańcuszkowej obroży psa też mógł nagle włączyć aparat.

Ultradźwięki zdolne są ponadto zakłócić pracę stymulatorów serca. W 1977 r. Japończycy wprowadzili sterowanie promieniami podczerwieni i ceny takich urządzeń zaczęły gwałtownie spadać. Obecnie wyposażenie jakiegokolwiek urządzenia w zdalne sterowanie kosztuje ok. 10 dolarów, a plastykowe pudełeczko może być droższe niż elektronika w środku.

WOZNIAK ZE ZDALNĄ KONTROLĄ

Amerykanie bogacą się. Obliczono, że na stoliku do kawy można zgromadzić nawet 11 pudełek. Już nie wiadomo, co lepsze: przebieranie w tym stosie czy wyszukiwanie jednego prawidłowego guziczka wśród kilkudziesięciu na jednej klawiaturze.

Steve Wozniak oferuje rozwiązanie wprowadzając skomputeryzowane zdalne sterowanie. Urządzenie jest wielkości dużego kalkulatorka kieszonkowego, którego pamięć może zapisać sobie właściwości impulsów potrzebnych do uruchomienia TV kablowej lub zwykłej czy magnetofonu, bez pomyłkowego włączenia wszystkiego na raz. Zawiera także zegar elektroniczny i budzik. Zdalna kontrola sięgnęła szczytów. Nawet nie trzeba być w domu, aby włączyć magnetowid i TV i nagrać sobie interesujący program. Inne zastosowanie to okresowe włączanie i wyłączanie różnych urządzeń, aby potencjalni włamywacze myśleli, że ktoś jest w domu.

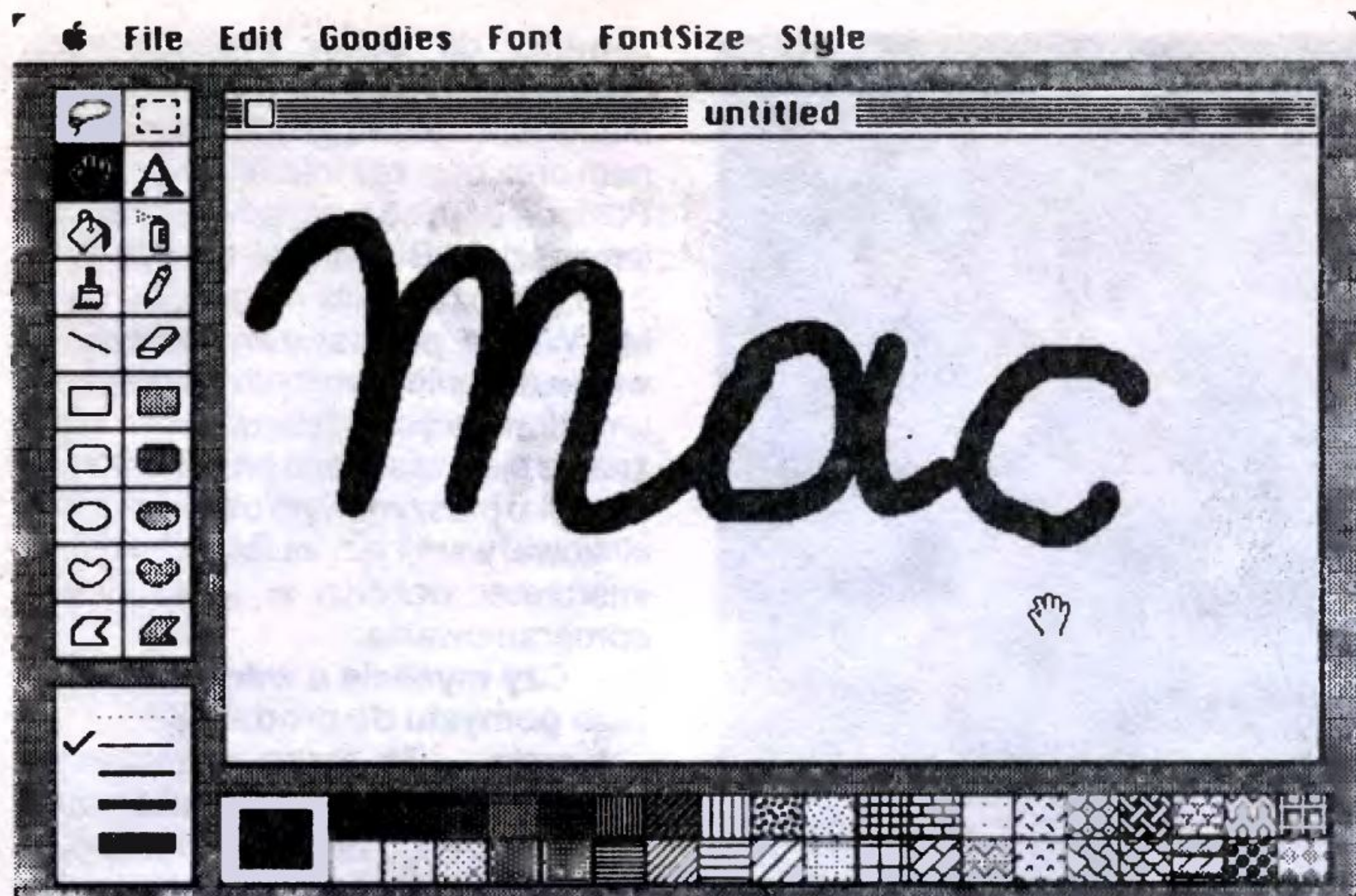
(JAL)

DZIECI KOCHAJĄ MYSZ

Parę miesięcy temu w mojej rodzinie pojawił się komputer. Bardzo szybko stał się przedmiotem niezbędnym nie tylko dla domowników. Używam go do pisania, obliczeń oraz do rysowania wykresów. Moi znajomi lubią grać w szachy, strzelać do desantu lub też pilotować prom kosmiczny. Te jednak zajęcia nie są dostępne dla moich dzieci – Iśki, która ma 8,5 roku oraz Ksanka, który ma 6,5. Większość bowiem zastosowań komputera osobistego zakłada umiejętność sprawnego czytania i pisania. Mimo to okazało się, że komputer stał się najbardziej ulubioną zabawką dzieci: w kąt poszły klocki Lego, dobranocki nikt nie ogląda, lalki i samochodziki kurzą się na półkach. Żona nie ma już problemu z niesforną dwójką. Za to czas spędzony przy komputerze ceniony jest wysoko. Stało się tak za sprawą myszy, właściwie należałoby powiedzieć, iż dzięki myszy oraz programowi rysującemu MACPAINT, który firma Apple dostarcza bezpłatnie każdemu posiadaczowi Maca (w naszym domowym języku Miękiszona). Dobrze wiedzieli, co czynią spece od marketingu, dając ten program za darmo i to każdemu, kto chce lub nie chce. Jestem przekonany, że w wielu domach (dotąd wyprodukowano już ponad 600 000 Macintoshy) trwa podobna wojna o prawo do zasiadania przed monitorem, z myszą w ręku. Choć monitor jest czarno-biały i ma tylko 9 cali przekątnej. A mysz ma tylko jeden przycisk. Nie przeszkadza to jednak erupcji talentu oraz kreatywności, jakiej doznaje każdy, komu dano szansę bawienia się programem rysującym. Zresztą określenie "program rysujący" nie jest tu chyba na miejscu. Lepiej byłoby powiedzieć, że jest to program, który pozwala każdemu zostać artystą. Proszę zresztą obejrzeć rysunki, reprodukowane w tym artykule. Zostały one narysowane przez dzieci, które nigdy przedtem nie miały do czynienia z komputerem, a i teraz traktują Maca raczej jako narzędzie rysujące niż komputer.

Spróbujmy opisać, dzięki czemu MACPAINT jest tak przyjemny w użyciu. Oto widok monitora, jaki wita rysownika po otworzeniu aplikacji (robi się to przyciskając raz po raz mysz, gdy kursor pokazuje piktogram MACPAINT).





psuje się, jeżeli coś źle zrobię – bardzo prędko, nawet u małych, cztero- czy pięcioletnich dzieci przechodzi. Prędko przekonują się, że warto próbować, a ponieważ efekty pracy można od razu obserwować na ekranie, więc następuje wzmocnienie pasji twórczej: rysuje się więcej i pewniej. Dla pedantów przewidziano możliwość wykańczania rysunków w modzie punktowym, czyli gdy rysunek jest powiększony, tak że widać poszczególne punkty.

Na koniec pytanie retoryczne: czy jest możliwe przeniesienie MACPAINTA na inne komputery. Istnieją już wersje na IBM PC (w dodatku także kolorowe, ale są kłopoty z kopiowaniem, bo nie ma dobrych i tanich drukarek kolorowych), są też podobne progra-



Narzędzia postawione pod ręką artysty komputerowego nie są wcale skomplikowane, a przy ich pomocy można narysować prawie wszystko. Najprostszym narzędziem jest ołówek, który rysuje cienką linię o dowolnym kształcie. Cienką czyli o grubości pojedynczego punktu. Uwaga! Mac ma ekran o rozdzielczości ok. 500 na 340 punktów, ale w MACPAINT można oglądać tylko małe okienko. Za to, używając łapki, możemy przesuwać okienko po typowej kartce formatu A4. W praktyce każdy rysuje w takim formacie, jaki mu najbardziej odpowiada. Swoboda twórcza sięga daleko!

Inne narzędzia reprezentowane są przez piktogramy, które tłumaczą się właściwie same: pędzel, którym można malować linie o różnej grubości, gumka do ścierania nieudanych partii rysunku, farba w rozpylaczu, którą można pokrywać części płaszczyzny. Mamy też pod ręką kubełek z farbą do wypełniania zamkniętych konturów. Jest to rodzaj wzoru, którym dokonuje się wypełnienie, wybiera się na dole okienka, można je też kreować samemu. Robienie podpisów jest banalnie proste: wybranie "A" powoduje umieszczenie napisu, który wprowadza się z klawiatury. Do rysowania kształtów zamkniętych (mogą one być automatycznie wypełnione wybranym wzorem) dysponujemy dużą liczbą narzędzi, wśród nich są gotowe ramki prostokątne oraz okręgi i elipsy.

Lasso oraz przerywana ramka pozwalają na wybieranie części rysunków, przemieszczanie ich, nakrywanie, powielanie. Dzięki akcesoriom właściwym wszystkim aplikacjom, jak wycinanie i zapamiętywanie wybranych kawałków, można bardzo łatwo przenosić rysunki. Wszystkie efekty pracy twórczej są oczywiście zapamiętywane na dysku, dzięki czemu można zawsze wrócić do zaczętego rysunku albo przerzucić fragment poprzedniego rysunku. By jeszcze bardziej uatrakcyjnić pracę, twórca programu MACPAINT (jest nim Bill Atkinson, jego podobizna zdobi okienko w czasie ładowania programu z dysku) przewidział rysowanie od razu w odbiciu lustrzanym lub też zamianę tła z rysunkiem.

Zapyta ktoś: mając do dyspozycji tyle narzędzi, młodzi artyści czują się pewnie zagubieni. Nieprawda! Wystarczy trochę cierpliwości, a samemu szybko dochodzi się do poznania coraz większej liczby przyrządów. Jest to nieodłączna część zabawy. Odkrycie, że farbą w rozpylaczu można pokrywać rysunki tak jak farbką wodną, to znaczy z prześwitywaniem tła, jest jednym z najwspanialszych odkryć w życiu komputerowego malarza. Potem następuje okres fascynacji. Aż do momentu znalezienia następnej możliwości, a jest ich tak dużo, że gwarantują przednią zabawę na długie zimowe wieczory!

Zaletą tak rozumianej grafiki komputerowej jest możliwość wydrukowania rysunku. Domowa wystawa prac malarskich jest nagrodą, pozwalającą w dodatku porównać, co kto potrafi. Zdarzają się czasami plagiaty, ale rysując "myszą" nigdy nie wykona się wiernej kopii. Takie więc plagiaty można potraktować jako pożyteczne rozszerzanie spojrzenia na świat.

Bardzo pouczająca jest obserwacja dzieci, które dopiero zaczynają poznawać MACPAINTA. Po krótkim okresie strach, że to komputer, że może zemy na SPECTRUM. Jednakże wymagania rozdzielczości oraz szybkości przetwarzania rysunków są tu tak duże, iż właściwie eliminują komputery 8-bitowe.

Czy programy malarskie trafią pod strzechy? Czyżby koniec z umazanymi buziami i ścianami?! Czy nowe tworzywo zastąpi plastelinę i pędzle? – Osobiście myślę, że raczej będzie to wspaniałe uzupełnienie tradycyjnych metod odwzorowywania rzeczywistości. A może ktoś napisze program, który będzie rzeźbił według komputerowego projektu? Patrząc na twórczość moich dzieci oraz ich zapał w komputerowym malowaniu, nie wątpię, że ich pokolenie inaczej będzie patrzyło na komputery i ich możliwości. Przede wszystkim opanuje lęk przed aparaturą elektroniczną i zdobędzie podstawy do prawdziwie twórczego, a potem praktycznego wykorzystania tego cudu techniki końca dwudziestego wieku.

Jakub Tatarkiewicz



OPOL-1 i co dalej?

Są młodzi, bardzo młodzi, jak "choroba", którą się zarazili. Jacek Domerecki, Marek Krokoszyński i Aleksander Sachanbiński jeszcze całkiem niedawno byli uczniami Zespołu Szkół Elektrycznych im. T. Kościuszki w Opolu. Za pracę pt. „Opracowanie dokumentacji technicznej i wykonanie mikrokomputera w oparciu o mikroprocesor Z-80" otrzymali wyróżnienie ubiegłorocznej edycji Turnieju Młodych Mistrzów Techniki organizowanego przez ZSMP.

Dwaj pierwsi postanowili dochować wierności komputerom. Będą studiować elektronikę w Politechnice Wrocławskiej. W nagrodę za OPOL-1 dostali przecież upragnione indeksy.

A wszystko zaczęło się tak. Szukali tematu do pracy dyplomowej. Ktoś rzucił im na zachętę: a może zrobicie mikrokomputer dydaktyczny? "Kupili" ten pomysł. Od września 1984 roku zaczął się młyn. Poszukiwanie i czytanie literatury, wycieczki po całej Polsce "za częściami", przygotowanie schematu, wyklejenie plansz obwodu drukowego płytki mikrokomputera, wykonanie klawiatury i obudowy. Pracowali po kilkanaście godzin dziennie, a przecież czekała ich też matura z innych przedmiotów. Ze swym konsultantem, inż. H. Białowieckim wyprzedzili znacznie program szkolny, poznając układy cyfrowe i technikę TTL. Radami praktycznymi służyli im także studenci i pracownicy Instytutu Cybernetyki Technicznej Politechniki Wrocławskiej oraz Zakładu Doświadczalnego "COBRAB" w Opolu, który finansował całe przedsięwzięcie. Za "ojca" całego przedsięwzięcia uważają swego opiekuna naukowego, profesora Rudolfa Zmarzłego. To on wskazał im drogę. Dzięki niemu "zachorowali" na komputery. Swoją pracę oceniają niezwykle skromnie. Po prostu przyłożyli się i coś zrobili, bo ich to wciągnęło. Podobnie skromnie, acz rzeczowo oceniają możliwości sprzętu, który zaprezentowali w szkole podczas obrony pracy dyplomowej.

Jacek: – Autorzy mikrokomputera OPOL-1 przyznają, że jest to bardzo proste urządzenie, o stosunkowo niskich parametrach technicznych. Z szerokiej gamy produkowanych na świecie mikroprocesorów wybrali jednak Z-80. Za nim przemawiały nie tylko dobre parametry, lecz także jego popularność i to, że jest produkowany w krajach socjalistycznych, co znacznie obniżyło koszty całego przedsięwzięcia.

Zbudowany przez nich mikrokomputer charakteryzuje się dużą prostotą obsługi i szeroką możliwością zastosowania do celów dydaktycznych. Może współpracować z dowolnym odbiornikiem telewizyjnym, magnetofonem oraz poprzęz interfejs z drukarką. Posiada pamięć programu ROM o pojemności 4 kB i pamięć danych RAM 2 kB, z możliwością rozbudowy do 16 kB. Wersja podstawowa pozwala na wprowadzenie prostych programów umożliwiających użytkownikowi zapoznanie się z zasadami programowania w języku maszynowym oraz stałoprzecinkowej wersji języka BASIC, którego interpreter wchodzi w skład stałego oprogramowania.

– Czy myślicie o wdrożeniu swojego pomysłu do produkcji?

Marek: – To raczej dalekosiężne plany, chociaż byłby to mikrokomputer dla każdego, kto rozpoczyna przygodę z elektroniką poprzez gry, zabawy, etc. ... Po przestawieniu go na język LOGO, doskonale nadawałby się dla dzieci, nawet 5-7-letnich. Pod jednym jednak warunkiem, że produkcja liczona byłaby w setkach tysięcy sztuk, a cena nie przekroczyłaby kilkunastu tysięcy złotych. Niestety, naszych obecnych sponsorów na to nie stać.

– Podobno złożyliście wniosek patentowy?

Jacek: – Układ wyświetlania, oparty wyłącznie na polskich elementach UCY-74 przy wykorzystaniu mikroprocesora, to nowoczesne rozwiązanie. Wniosek zgłosiliśmy przez szkołę, lecz rzecznik patentowy nas zniechęcił, twierdzac, że procedura potrwa 2-4 lat. Zdajemy sobie sprawę, że po tym czasie nasze rozwiązanie będzie niewiele warte, zwłaszcza że w mikroelektronice postęp jest bardzo szybki. I chociaż w Polsce myśl techniczna i jej produkty są coraz lepiej widoczne, to jednak dostęp do informatyki nie jest ciągle powszechny. Już za kilka lat grozi nam analfabetyzm informatyczny. Sytuację może uratować jedynie naprawdę popularny, tani mikrokomputer kosztujący ok. 20 tys. złotych. Tymczasem np. w szkołach często brakuje magnetofonów. Na telefon czeka się bez mała 20 lat; cóż więc mówić np. o sieciach informatycznych. Ta technika wymaga 100-procentowej dokładności przy produkcji i programowaniu, a więc zmiany metod pracy, sposobów myślenia...

Marek: – Jeżeli przemysłowi naprawdę zależałoby na osławionych "trzech S", to z konieczności sięgnąłby po mikroinformatykę. Wygląda na to, że nie zależy nam na produkcji opłacalnej, nowoczesnej i konkurencyjnej. Komputery nie mogą pracować w przedsiębiorstwach, w których panuje bałagan. Poza tym wielu ludzi straciłoby pracę, a przynajmniej musiałoby ją zmienić. Dla "świętego spokoju" unika się więc komputerów.

Jacek: – A ja uważam, że jesteśmy młodym społeczeństwem, i to gwarantuje, że nacisk na wprowadzenie i powszechne stosowanie mikrokomputerów będzie coraz silniejszy. To po prostu konieczność.

Michał Liszka

JAKI BĘDZIE RYNEK KOMPUTERÓW 1986

W amerykańskiej branży mikrokomputerów zapanowało obecnie pewne zniechęcenie. Po wzroście sprzedaży o 30% rocznie w 1983 r. i pierwszej połowie 1984 r. popyt (dla wielu nieoczekiwanie) spadł, i to znacznie. W ubiegłym roku według wstępnych szacunków sprzedano ich zaledwie o 3% więcej niż w r. 1984.

Według genialnych konstruktorów wspomaganych przez błyskotliwych handlowców są nadzieje na ożywienie popytu i zwiększenie sprzedaży w 1986 r., bowiem zdumionemu światu zamierza się oferować supermikrokomputery i rewelacyjne oprogramowanie. Liczy się w tych kołach, że mało kto oprze się pokusie posiadania nowego, najnowocześniejszego cacuszka. Ale w USA skonfrontowano hurraoptymistyczne oceny niektórych firm z paroma faktami ekonomicznymi. Reprezentatywnymi ankietami przebadano plany zakupów sprzętu informatycznego firm handlowych, przemysłowych, usługowych, finansowych itp. i okazało się, że tempo wzrostu na pewno nie będzie dwucyfrowe, bowiem rynek jest w pewnym sensie nasycony. Liczba pracujących w wielu firmach mikrokomputerów potroiła się w ciągu paru lat i teraz wszyscy uczą się nowego stylu pracy z nowym sprzętem i różnymi oprogramowaniami.

Poza tym mikrokomputery straciły charakter inwestycji wyjątkowej. Coraz powszechniej traktuje się je jak zwykłe inwestycje. Kiedyś w XIX wieku tymi wyjątkowymi inwestycjami były koleje, a na początku naszego stulecia – maszyny ciężkie. Mikrokomputery, jak normalne dobra inwestycyjne, zaczęły podlegać cyklom koniunkturalnym. Tymczasem w USA od 1983 r. nakłady inwestycyjne systematycznie zmniejszają się. Wprawdzie ankietowani dyrektorzy ds. informacji w przedsiębiorstwach użytkujących mikrokomputery mówili o zwiększeniu nakładów na nie o 11%, ale ich dyrektorzy naczelni – już tylko o ogólnym wzroście funduszu inwestycyjnego o 3% (w bieżących dolarach). Na tej podstawie eksperci szacują, że w USA zakupi się o 5% więcej sprzętu informatycznego, przypominając zarazem, że tempo to będzie i tak o wiele szybsze niż ogólny wzrost inwestycji.

W Polsce można jeszcze mówić o dominacji osób prywatnych jako kategorii nabywców, ale na 5 mikrokomputerów sprzedawanych w USA, 4 kupowane są przez firmy. Podobnie jest w innych krajach uprzemysłowionych. Rodzi to dla nas nadzieje, że producenci, nie mogąc upchnąć ich u siebie, będą dążyć energiczniej niż dotąd do obejścia w jakiś sposób embarga na eksport komputerów do krajów socjalistycznych przez COCOM, i dzięki temu na naszym rynku pojawi się, po przystępnych cenach, więcej urządzeń dobrych, choć na pewno nie tych najnowocześniejszych.

Także w Europie Zachodniej największe firmy komputerowe mają trudności. Popyt na układy scalone w 1985 r. znacznie się zmniejszył. Choć zamówienia na układy scalone powoli wzrastają, mało kto ocenia sytuację na rynku sprzętu na tyle optymistycznie, aby robić zapasy kostek. Perspektywy dla branży elektronicznej nie są różowe. Według specjalistów z W. Brytanii, z obecnych 17 firm przetrwa rok bieżący zaledwie pół tuzina.

(JAL)

KOMETA HALLEYA

Szanowna Redakcjo,
przesyłam program w języku Logo na temat całkiem aktualny do ewentualnego opublikowania na łamach Waszego interesująco zapowiadającego się i bez wątpienia potrzebnego pisma.

Załączony program daje niepowtarzalną, jedyną w swoim rodzaju, okazję obejrzenia na ekranie telewizora (najlepiej kolorowego) komety Halleya!!! Jedyne co do tego potrzebne, to – oprócz programu: mikrokomputer ZX Spectrum, translator Logo i ...odrobina wyobraźni (a tę posiadają i duzi, i mali – sprawdziłem doświadczalnie).

Zważywszy, że w kwietniu nastąpi zbliżenie komety Halleya do Ziemi, a pogoda niespecjalna – Redakcjo! nie trać szansy! Tylko u nas...!

Zdzisław Płoski
Instytut Informatyki UWr.

```
TO KOMETA
HT CS PD
SETBG 0 SETBR 0 SETTC [0 6]
INTRODUKCJA
CS
SETCUR [10 3]
PR [PER COMPUTER]
SETPC 7 KROPKI.
SETCUR [12 19]
PR [AD ASTRA]
SETPC 6 KROPKI...
SETTC [0 4] SETCUR [14 21]
PR [©ZPL]
MAKE "x -90 + RANDOM 45
MAKE "y -30 + RANDOM 15
SETPC 4 KROPKI
END
TO INTRODUKCJA
SETCUR [10 2] PR [KOMETA © ZPL]
SETCUR [3 7]
PR [Program startuje po]
PR [naciśnięciu dowolnego klawisza.]
PR [] PR []
PR [UWAGA: W trzeciej minucie]
PR [działania programu]
PR [widoczna kometa Halleya!]
WAIT 120
MAKE "start RC
END
TO KROPKI.
REPEAT 50 + RANDOM 100
DOT SE (RANDOM 120) - 60
(RANDOM 50) - 25
END
TO KROPKI..
REPEAT 100 + RANDOM 200
DOT SE (RANDOM 180) - 90
(RANDOM 76) - 38
END
TO KROPKI
REPEAT 10 + RANDOM 10
DOT SE (RANDOM 240) - 120
(RANDOM 100) - 50
SETPC 7
DOT SE :x + RANDOM 7:(y) +
RANDOM 4
SETPC 4 + RANDOM 2
MAKE "x :x + RANDOM 1 +
RANDOM 2
MAKE "y:y + RANDOM 2
IF:y83*) TOPLEVEL
KROPKI
END
```

*) albo 47.

DZIKIE

SHERRY TURKLE

Pewnego zimowego stycznioweg0 poranka roku 1800 z lasów położonych koło wioski Saint-Sernin w okręgu Aveyron na południu Francji wyłonił się chłopiec w wieku około 13 lat. Nikt nie wiedział, skąd się tam znalazł. Wszystko wskazywało na to, iż od wczesnego dzieciństwa żył samotnie. Nie potrafił mówić, wydawał tylko dziwne, nic nie znaczące okrzyki.

Dzikie Dziecko było człowiekiem, aczkolwiek żyło w izolacji od kultury i języka. Jego życie stało się tym co nazwano "zakazanym doświadczeniem", doświadczeniem, które pokazałoby czym naprawdę jest istota ludzka pozbawiona otoczki społeczeństwa i kultury. Młody francuski lekarz, Jean-Marc-Gaspard Itard usiłował uczyć Dzikie Dziecko, któremu dano imię Wiktor, ale nawet po siedmiu latach wszechstronnego, systematycznego, natchnionego nauczania chłopiec nie nauczył się ani mówić, ani czytać, ani pisać.

Choć eksperyment niczego nie wyjawił, historia Dzikiego Dziecka nie straciła swej magii. Ludzie mogli stawiać siebie w takiej sytuacji, mogli zadawać sobie pytania: Jestem Itardem. Moim zadaniem jest uczenie Dzikiego Dziecka. Czego powinienem spróbować? Co i dlaczego może się wydarzyć? W czasie tego eksperymentu myślowego ujawniały się ich wyobrażenia o tym, kim jest człowiek i w jaki sposób się rozwija.

Nie jest niespodzianką, że historia Dzikiego Dziecka została ostatnio ponownie odkrytą. Pojawiło się mnóstwo nowych opracowań na ten temat: historycznych, literackich, psychologicznych. Historia ta nadal inspiruje i świetnie nadaje się do głębokich przemyśleń. Ostatnio pojawiło się jednak coś nowego. Nowy przedmiot zakazanego eksperymentu. Nowa forma rozumu, która właściwie nie jest jeszcze rozumem. Coś pośredniego, równie podległego nauce jak przesądom. Jest to komputer.

Chcieliśmy, aby Dzikie Dziecko wyjawiło nam powiązanie między nami i naturą. Od komputera żądamy czegoś więcej. Chcemy wiedzieć nie tylko jaka jest nasza pozycja w naturze, ale również co wiąże nas ze światem naszych wytworów.

Komputer tworzy człowieka

Podręcznikowa historia rozwoju techniki skupia się głównie na zagadnieniach praktycznych. W tym ujęciu zbudowanie teleskopu wiodło bezpośrednio do odkrycia nowych gwiazd, budowa kolei do zdobycia nowych obszarów. Jednocześnie jednak nowe miejsce Ziemi w systemie słonecznym zmusza nas do przemyślenia naszego stosunku do Boga; możliwość przemierzenia kontynentu w ciągu kilku dni oznacza nowe pojęcie odległości i komunikacji. Zegary pozwoliły na coś więcej niż dokładne mierzenie czasu – uczyniły czas czymś "podzielnym" i abstrakcyjnym. Czas przestał się wiązać z wykonaniem jakiejś pracy, przestał być powiązany z ruchem słońca lub księżyca czy też zmianą pór roku. Stał się tym, czym jest przesunięcie się wskazówek na tarczy zegara. Nasze pojęcie czasu jeszcze raz zostało zmienione wraz z pojawieniem się czasomierzy cyfrowych. Czas stał się jeszcze bardziej abstrakcyjny. Przestał być procesem, a zaczął być informacją.

Zmiany technologiczne przyspieszają nie tylko zmiany w tym, co robimy, ale i w tym, jak myślimy. U ludzi zwiększa się poczucie własnego istnienia, istnienia innych ludzi i ich powiązań ze światem. Nowe urządzenie skryte wśród migających sygnałów cyfrowych, w odróżnieniu od zegara, teleskopu, czy pociągu, jest urządzeniem, które "myśli". Zmusza do zmiany nie tylko naszego pojęcia czasu lub odległości, ale również do zmiany pojęcia rozumu.

Większość rozważań na temat komputerów koncentruje się na instrumentalnej stronie zagadnienia, na tym co komputery będą robiły. Moja uwaga tutaj jest skupiona na czymś innym, na komputerze "podmiotowym". Na tym, czym jest wprowadzenie tego urządzenia w życie społeczne i rozwój psychologiczny; na tym, jak komputer wpływa na nasz sposób myślenia, szczególnie na to, co myślimy o samych sobie. Rzeczą, która mnie fascynuje jest nie postawione pytanie, skryte pod większością naszych rozważań na temat możliwości komputerów. Pytanie nie o to, jakie będą w przyszłości komputery, ale jacy my będziemy? Jakimi ludźmi się stajemy?

W większości opracowań komputer jest opisywany jako urządzenie zunifomizowane, racjonalne i ściśle uwarunkowane logiką. Moje spojrzenie na komputer jest inne, nie rozważam istoty jego budowy, a raczej jego "drugą naturę" jako obiektu inspirującego, obiektu, który fascynuje, zakłóca spokój ducha i pobudza myślenie.

Komputery wywołują silne uczucia, nawet u tych, którzy z nimi nie stykają się bezpośrednio. Ludzie wyczuwają w nich obecność czegoś nowego i ekscytującego, jednak głównie obawiają się ich mocy, a także zagrożenia z ich strony. Czytają gazety, w których pisze się o "wdowach komputerowych" i "komputerowych nałogowcach". Rodzice są zaniepokojeni zainteresowaniem swoich dzieci nie tylko komputerami, ale i młodszymi siostrami i braćmi tych urządzeń – nową generacją zabawek elektronicznych. Dzieci, które nigdy nie siedziały spokojnie nawet przed ekranem telewizora, potrafią być całkiem pochłonięte taką zabawką. Rodzice zauważają, że zabawki te mogą być pomocne w nauczaniu czy nauce, lecz obawiają się stopnia zaangażowania uczuciowego dzieci. "Niepokojące jest, kiedy ich kolegami są maszyny". "Nie chciałabym, aby mój syn zabierał swój komputer do łóżka. Nie miałabym nic przeciwko książce, jakieś zwierzątko nawet by mnie ucieszyło – ale zabieranie maszyny do łóżka wywołuje u mnie mieszane uczucia". Siedzę na ławce w parku z matką 6-letniej dziewczynki, bawiącej się w pytania i odpowiedzi z zabawką sterowaną komputerowo. Dziecko odpowiada maszynie, gdy ta upomina je za złą odpowiedź, lub gratuluje prawidłowej. Mój Boże – wzdycha matka – ona traktuje zabawkę jak osobę. Czy uważa pani, że również ludzi traktuje jak maszyny?

Na przykładzie tej matki widać szok pierwszego zetknięcia się z komputerem. Ale kontakt z komputerem w jeszcze znacznie większym stopniu wpływa na tych, którzy znają go dobrze i działają na nim bezpośrednio, którzy mają tę możliwość, by odczuć jego "drugą naturę".

Zwierciadło duszy

To właśnie oni podkreślają "przyciągającą siłę" komputerów. Stwierdzają, że są to urządzenia fascynujące, trudne do odstawienia. Dla niektórych ta przyciągająca siła jest źródłem zagadkowej przyjemności: dla prawnika z jednej z firm na Wall Street, gdzie zainstalowano system komputerowy, który nauczył się samodzielnie obsługiwać komputer, by móc korzystać z zawartych w nim informacji, komputer stał się czymś pośrednim pomiędzy krzyżówką w niedzielnej gazecie a kostką Rubika. U innych uczucia związane z komputerami są bardziej intensywne,

DZIECKO

bywają nawet groźne. Uważają, że znajdują się pod wpływem siły, której wpływ i bliskość przewyższają wszystko, czego dotychczas zaznali. Wiele osób, od natchnionych programistów poczynając, a kończąc na tych, których kontakt z komputerem ogranicza się tylko do gier, porównuje swe doświadczenia z komputerami z seksem, narkotykami lub transcendentalną medytacją. Zdolność reakcji i złożoność komputera wpływają na pewną swobodę w jego opisie.

Narzuca się pewna analogia: komputer podobnie do testów Rorschacha z atramentem i kleksami jest świetnym środkiem do projekcji osobowości.

Komputery stają się przedmiotami powszechnymi w codziennym życiu – w rozrywce, nauce oraz pracy – każdy więc będzie miał możliwość zetknięcia się z nimi w sposób, w którym maszyna może służyć jako swoiste zwierciadło duszy.

Gdy różni ludzie siadają do komputera, nawet by wykonać to samo zadanie, ich styl współpracy z komputerem jest różny. Najłatwiej to zauważyć w trakcie programowania. Wielu odbiera programowanie jako tworzenie oddzielnego świata. Niektórzy tworzą światy, których działanie jest łatwo przewidzieć, podbudowując tym samym w sobie poczucie możności sprawowania ścisłej kontroli nad nimi. Inni, mając inne potrzeby i pragnienia, tworzą swe światy w sposób tak skomplikowany, że są one na granicy wymknięcia się spod kontroli.

Oczywiście istnieje różnica pomiędzy komputerem a testami Rorschacha – kleksy pozostają na papierze, a komputer staje się częścią codziennego życia. Może być zarówno środkiem do tworzenia, jak i odtwarzania. Gdy tworzysz programowany świat, to pracujesz w nim, eksperymentujesz w nim, żyjesz w nim. Ta ogromna elastyczność komputera, fakt, że poprzez stworzenie oprogramowania staje się on nową istotą, czyni go idealnym środkiem do budowy całej gamy prywatnych światów, a poprzez nie do lepszego poznania samego siebie. Komputer to coś więcej niż tylko ekran do projekcji osobowości – wcielił się on w proces dorastania nowych pokoleń. Dla dzieci i dorosłych, którzy używają komputerów do gier, obróbki tekstów, informacji, obrazów, a szczególnie dla tych, którzy uczą się programowania, wpływają one na rozwój osobowości, tożsamości a nawet seksualizmu.

Komputer żyje?

Doświadczenia z komputerami stają się więc punktami odniesienia dla przemyśleń i dyskusji na temat edukacji, społeczeństwa, polityki, a przede wszystkim, co najważniejsze w tej książce, na temat natury człowieka. W tym ujęciu komputer jest "maszyną metafizyczną". Również i dzieci są pobudzane. Komputer stwarza jeszcze jedną okazję do zadania sobie podstawowych pytań, na które dzieciństwo musi znaleźć odpowiedzi, a pomiędzy nimi pytania: „Co to jest życie?"

W świecie dorosłych specjaliści wiodą spory o to, czy komputer może uzyskać rzeczywistą "sztuczną inteligencję", czyli zdolność do samodzielnego, podobnego człowiekowi, myślenia. Jednakże niezależnie od inteligencji przyszłości, komputer już obecnie wpływa na sposób myślenia dzieci i tworzenie przez nie takich pojęć jak żywy i nieożywiony, mający i nie mający świadomości.

Niektóre przedmioty, a w naszych czasach komputer jest wśród nich najbardziej znaczącym, skłaniają do przemyślenia podstawowych reguł i zasad. Dzieci bawiące się zabawką, co do której mają wrażenie, że jest żywa, i dorośli igrający z wytworem myśli zawartym w programie, są przyciągnięci przez komputer poprzez jego zdolność do inspiracji i zdolność do ubarwienia wewnętrznych przemyśleń. Komputer jest maszyną "metafizyczną", "psychiczną" nie tylko dlatego, że posiada swą specyficzną psychologię, ale dlatego, że wpływa na nasz sposób wyobrażania sobie nas samych.

Studia nad komputerami i ludźmi rozpoczęłam w MIT sześć lat temu. Uderzyły mnie psychologiczne debaty na temat komputerów oraz stopień, do jakiego moi koledzy i studenci korzystali z psychologii przy opisie zjawisk komputerowych. Nie działał program szachowy. Jego twórcy opisywali to w następujący sposób: "Gdy czuje się zagrożony, zaatakowany, wówczas dąży do wysunięcia króla. Mylą mu się wartość i siła, co prowadzi do samozniszczenia". Nawet najbardziej techniczne dyskusje na temat komputerów zawierają terminy zapożyczone z dziedziny opisu ludzkiego sposobu myślenia. W słowach swoich twórców, programy mają intencje, chcą być najlepsze, są mniej czy bardziej mądre, czy głupie, porozumiewają się między sobą i bywają zmieszane. Ten psychologiczny słownik nie powinien być zaskakujący. Wiele osób uważa komputery za swoiste obiekty matematyczne, jednakże gdy się do nich zbliżyć, jasnym staje się, iż są to obiekty informacyjne przetwarzające symbole i używające języka. Nieuniknionym jest stwierdzenie, że współpraca z komputerem jest tak jak współpraca z żywym umysłem, nawet jeśli trochę ograniczonym. Dlatego właśnie język, który ukształtował się w środowisku komputerowym, jest specyficzny. Żargon komputerowy w pewnym sensie jest swoiście "rozumowy". Co więcej, nie tylko o komputerze myśli się w ludzkich kategoriach rozumowych – istnieje również odwrotne zjawisko. Ludzie coraz częściej myślą o sobie w kategoriach komputerowych.

Czy pani ma bufor?

Naukowiec zajmujący się komputerami stwierdza: "Mój następny wykład ma realizację sprzętową", rozumiejąc przez to, że może on być wygłoszony bez specjalnego zaangażowania myśli, czy też gdy nie chce, aby mu przerwano w czasie zażartej dyskusji przy stole, stwierdza, że "musi opróżnić bufor". Inny naukowiec psychoterapię określa jako "odpluskwienie" – technikę stosowaną do ostatecznej korekty uruchamianych programów, oraz określa jej "wartości domyślne" przy zastosowaniu do mężczyzn.

Ludzie ci używają gwary komputerowej nie tylko jako pewnego sposobu mówienia. W ich sposobie formułowania myśli jest zawarte bezpośrednie porównanie procesów zachodzących w psychice człowieka do procesów zachodzących w maszynie. Taka formuła sugeruje, że my również jesteśmy systemem informacyjnym, gdzie myśli są tworzone przez "sprzęt", gdzie istnieją bufory, które muszą być opróżnione przed otrzymaniem nowej porcji informacji w kontaktach z innymi ludźmi, że dla każdego zadania istnieją pewne wartości domyślne, do których zawsze możemy się odwołać i że problemy emocjonalne są błędami, które możemy wyplenić.

"Realizacja sprzętowa", "bufor", "wartość domyślna", "odpluskwić" – to niektóre z wyrażeń komputerowych, z którymi zetknęłam się w kulturze komputerowej MIT. Inne, które pojawiły się wcześniej, weszły już na dobre do języka potocznego, jak na przykład dobrze znane pojęcie programowania. Na bardzo wczesnym etapie pisania tej książki, w czasie obiadu, który spożywałam wspólnie z jednym z moich przyjaciół, usiłowałam wyjaśnić mu zachodzący proces rozszerzania się zasięgu pojęć komputerowych. Problem rozwiązał się sam, gdy przy sąsiednim stoliku usiadły dwie młode kobiety i jedna powiedziała do drugiej: "Najtrudniej przeprogramować się na życie w samotności". Język komputerowy tak bardzo rozszerzył swe znaczenie, że czasami zapominamy o źródłosłowie poszczególnych wyrazów. Jednakże, mimo że potrafimy zapomnieć, to nie unikniemy przyjęcia nowych założeń, które niesie ze sobą język, określających kim jesteśmy i jak możemy się zmieniać. Komputer znalazł swe trwałe miejsce w kulturze jako całości.

tłumaczył:
Leszek Kamionka

O KOMPUTERACH
I NIESKOŃCZONOŚCI

Szanowny Panie Redaktorze!

Skreśliłem te parę słów do Pana w nadziei, że przyjmie Pan życzliwie moją pomoc w rozumieniu niektórych problemów związanych z komputerami. To, o czym piszę, nie jest wynikiem obecnego dynamicznego rozwoju informatyki i komputeryzacji. Zrodziło się w głowach matematyków wtedy, gdy jeszcze nie było elektronicznych urządzeń liczących. Wynikło natomiast z rozważań nad istotą matematyki.

Wśród tych rozważań pojawiło się pytanie, które można postawić właśnie teraz, gdy komputer stoi na biurku. Czy zastanawiał się Pan nad tym, co da się policzyć za pomocą komputera? Oczywiście nie myślę o żadnym konkretnym komputerze. Mam na myśli symboliczną maszynę liczącą. Słowa "policzyć" też użyłem w symbolicznym sensie. Wiadomo przecież, że jednostka centralna w każdym komputerze działa tylko na liczbach naturalnych reprezentowanych jako ciągi zer i jedynek. Dopiero w zależności od kontekstu dana liczba naturalna, ukryta w pamięci, jest interpretowana jako instrukcja, część większej liczby czy też wskaźnik pewnego stanu maszyny. Na skutek realizacji programu wewnątrz maszyny też pojawiają się tylko liczby naturalne. My otrzymujemy napisy i obrazki na ekranie dzięki odpowiedniej interpretacji tych liczb przez urządzenia zewnętrzne, monitor czy drukarkę.

W świetle tego wyjaśnienia wydaje się zasadne patrzeć na komputer jak na maszynę, która może znajdować wartości pewnych funkcji dla pewnych argumentów będących liczbami naturalnymi. Oczywiście wartości tych funkcji też są liczbami naturalnymi. Pytanie sprowadza się więc do następującego problemu: "Jakie funkcje o argumentach i wartościach naturalnych może policzyć komputer?"

Odpowiedź nie wydaje się łatwa. Zacznijmy więc od najprostszych funkcji. Dla nikogo chyba nie jest tajemnicą, że komputer może obliczyć funkcję stale równą zero, może też do dowolnej liczby naturalnej dodać jeden. Podobnie zgodzimy się z tym, że tzw. funkcja rzutowania też da się obliczyć za pomocą komputera (funkcja rzutowania jest funkcją wielu zmiennych, która wybiera jedną spośród danego ciągu liczb; np. rzutowanie na piątą zmienną z dziewięciu, dla dziewięciu liczb będących argumentami, daje w wyniku wartość równą piątej liczbie).

Jestem przekonany, że potrafi Pan napisać algorytm obliczania każdej z tych funkcji dla dowolnych liczb naturalnych, a także zrealizować ten algorytm na dowolnym z komputerów. Myślę, iż nie muszę Pana przekonywać, że jeżeli można obliczyć za pomocą elektronicznej maszyny liczącej funkcje f, g i h, to można za pomocą tej samej maszyny policzyć wartość złożenia tych funkcji. Znaczy to, że mamy policzyć wartość funkcji f na argumentach będących wartościami funkcji g i h. Takie obliczenie wykonamy kolejno: najpierw obliczymy wartość funkcji g i przechowamy tę liczbę w pamięci. Następnie obliczymy funkcję h i w końcu wykonamy obliczenie funkcji f, ale nie dla danych wejściowych, lecz dla wyliczonych wcześniej wartości funkcji g i h.

Trochę trudniej wykonać obliczenie wartości operatora μ – najmniejszej liczby spełniającej pewien warunek. Zastosowanie operatora μ do funkcji f może mieć miejsce tylko wtedy, gdy jest taka liczba naturalna n, że f(n) jest zerem. Wtedy μf jest najmniejszą liczbą naturalną, dla której funkcja f przyjmuje wartość 0. Wyobrażam sobie, że jeżeli potrafimy obliczyć wartość funkcji f dla każdej liczby naturalnej, to wykonujemy obliczenia po kolei: najpierw dla zera, potem dla jedynki, dwójki i tak dalej. Obliczenia kończymy natychmiast, gdy wyliczona wartość będzie równa zero. Takie postępowanie musi kiedyś się zakończyć, gdyż założyliśmy, że dla pewnej liczby naturalnej n, f(n) =0, czyli musimy sprawdzić co najwyżej n wartości. Można również na komputerze (wyposażonym w odpowiednie „narzędzia") wykonać obliczenia rekurencyjne. To znaczy, że jeżeli wiemy jak obliczyć pewną funkcję h, to możemy zdefiniować nową funkcję f podając tylko jej wartość w zerze f(0) = k oraz deklarując, że f(n+1) = h(n,f(n)). Jak widać, do obliczenia wartości funkcji f na liczbie n musimy wyliczyć wcześniej wartość tej funkcji dla liczby poprzedzającej n.

Zaczyna się Pan pewnie zastanawiać, po co o tym piszę. Co mają wspólnego takie proste funkcje i operacje na nich z olbrzymimi możliwościami komputerów, z nieprzebranymi zasobami mocy obliczeniowej wielkich maszyn liczących? Otóż okazało się – i to dość dawno – że wszystko, co można obliczyć za pomocą komputera, a nawet dużo więcej: wszystkie funkcje obliczalne przy pomocy algorytmów w najszerszym intuicyjnym sensie są funkcjami rekurencyjnymi. Funkcje rekurencyjne zaś to funkcje określone na podzbiorach zbioru liczb naturalnych, które można otrzymać z funkcji elementarnych (dodawania, mnożenia, rzutowania i porównywania) za pomocą składania i operatora μ. Dokładniej: funkcja f jest funkcją rekurencyjną, jeżeli istnieje ciąg funkcji $f_1, f_2, ... f_n$ taki, że $f_n = f$, zaś każda funkcja f_i z tego ciągu jest funkcją elementarną, bądź powstała za pomocą złożenia lub operatora μ z funkcji o numerach mniejszych niż i.

To ważne zdanie, które sformułowałem wyżej, zostało wypowiedziane w 1936 roku przez Churcha i jest znane jako teza Churcha. Nie mogę, niestety, podać dowodu tej tezy, gdyż dowód taki nie istnieje. Nie może istnieć, ponieważ teza porównuje z jednej strony klasę funkcji bardzo dokładnie matematycznie opisaną, tzn. klasę funkcji rekurencyjnych, z drugiej zaś strony – klasę opisaną tylko intuicyjnie, bez żadnej precyzji. Tym niemniej teza Churcha wydaje się być mocno uzasadniona. Analiza wielu algorytmów wykazuje także zgodność z tezą Churcha.

Zgodzę się z Panem, jeżeli będzie Pan twierdził, że w definicji funkcji rekurencyjnych nie widać związku z maszynami liczącymi. Popatrzmy więc na problem funkcji obliczalnych przez komputery od innej strony. Przeanalizujmy pracę człowieka przy znajdowaniu wyniku dla pewnej funkcji obliczalnej, tzn. zapisanej za pomocą algorytmu. Biegły rachmistrz wypisuje najpierw wszystkie dane. Potem spogląda na przepis postępowania – algorytm – i wykonuje pierwszą zapisaną czynność. Musi więc zmodyfikować w pewien określony sposób dane początkowe. Gdy już się z tym upora, realizuje następne polecenie itd. Za każdym razem otrzymuje nowy ciąg symboli, który znów modyfikuje, aż do wykonania wszystkich czynności przewidzianych algorytmem. Taka analiza postępowania rachmistrza doprowadziła matematyków do stworzenia teoretycznego modelu abstrakcyjnej maszyny liczącej. Jest to urządzenie, które może modyfikować w pewien zadany sposób swoją pamięć wewnętrzną. Modeli takich powstało kilka.

Pozwolę sobie przedstawić jeden z nich, zaproponowany w połowie lat trzydziestych przez A.M. Turinga.

Wygodnie jest wyobrazić sobie maszynę Turinga jako czarną skrzynkę wyposażoną w nieskończoną pamięć wewnętrzną – taśmę – i w ustalony skończony zbiór stanów wewnętrznych. Wśród stanów wewnętrznych mamy dwa wyróżnienia – stan początkowy i końcowy. Taśma jest podzielona na klatki i nieograniczona w prawo i w lewo. W każdej klatce taśmy można zapisać jakiś symbol z ustalonego skończonego alfabetu. Maszyna Turinga pracuje skokowo w tzw. czasie dyskretnym. W każdym momencie czasu pracy maszyna Turinga znajduje się w jednym ze swoich stanów wewnętrznych i testuje jedną klatkę taśmy – rozpoznaje zapisany w niej symbol. Jeżeli maszyna Turinga nie jest w stanie końcowym, to w następnej chwili wykona pewną operację. Może zetrzeć symbol z taśmy, zapisać inny, przesunąć taśmę w lewo lub w prawo o jedną klatkę, wreszcie zmienić swój stan wewnętrzny. To, którą z tych operacji wykona, jest zapisane w programie maszyny Turinga i wyznaczone jednoznacznie przez aktualny stan maszyny i symbol w testowanej klatce taśmy.

Dla lepszego wyobrażenia sobie działania maszyny Turinga proszę porównać ją z jakimś procesorem. Stan wewnętrzny możemy interpretować jako licznik rozkazów. Wtedy zmiana stanu maszyny to po prostu przejście do innego rozkazu, czy też wykonanie skoku. Przesunięcie taśmy to zmiana wskaźnika pamięci, a zapisywanie symbolu w klatce taśmy – to wykonanie rozkazu zapisu do pamięci.

W rozważaniach teoretycznych maszynę Turinga często utożsamia się z jej programem. Program dla maszyny Turinga musi zawierać dla każdej pary – stan wewnętrzny i symbol na taśmie – opis wykonywanej akcji i stan wewnętrzny, w którym znajdzie się maszyna po wykonaniu opisanej operacji. Jeżeli mamy daną maszynę Turinga, to wartość liczonej przez nią funkcji znajdujemy następująco. Na taśmie umieszczamy słowo w naszym alfabecie, tzn. pewien ciąg symboli. Jest to dana wejściowa. Wprowadzamy urządzenie w stan początkowy, ustawiamy taśmę na początek naszego słowa i startujemy maszynę. W kolejnych krokach maszyna będzie zmieniała nasze słowo, wstawiając nowe symbole, ścierając stare. Będzie też przesuwała taśmę. Jeżeli maszyna wreszcie dojdzie do stanu końcowego to słowo, które zostanie na taśmie, uznamy za wartość liczonej funkcji. Jeżeli maszyna Turinga nigdy nie dojdzie do stanu końcowego, to znaczy, że funkcja nie jest określona na zadanym słowie.

Myślę, że podobieństwo między realnym komputerem a maszyną Turinga jest tak duże, iż możemy łatwo uwierzyć, że to, co da się poobliczać za pomocą komputera, można policzyć pewną maszyną Turinga.

Uprzedzając Pańskie obiekcje powtórzę. Przeanalizowano wszystkie znane algorytmy (te, które działają na komputerach, i te, które jeszcze nie mają swojej implementacji) i stwierdzono, że każdy taki algorytm da się przedstawić za pomocą pewnej maszyny Turinga. Odwrotnie oczywiście nie jest – nie wszystko, co da się obliczyć pewną maszyną Turinga, można zrealizować na komputerze. Przeszkadza tu założenie o nieskończoności taśmy w maszynie Turinga. Tym niemniej możemy stwierdzić (powtarzając tezę Turinga), że wszystkie funkcje obliczalne przez algorytmy w intuicyjnym sensie da się obliczyć maszynami Turinga. Tutaj znów nie można przedstawić ścisłego dowodu, z powodów takich jak przy tezie Churcha.

Z tezy Churcha i tezy Turinga wynika, że funkcje obliczalne przez maszyny Turinga to dokładnie funkcje rekurencyjne. Oba pojęcia są precyzyjnie matematycznie zdefiniowane, a więc powinna dać się udowodnić ich równoważność. I tak jest w istocie! Można podać precyzyjny dowód, że dla każdej funkcji rekurencyjnej istnieje maszyna Turinga, która liczy tę funkcję, i na odwrót, że funkcje liczone przez maszyny Turinga są rekurencyjne.

Tak więc doszliśmy do końca naszych rozważań. Warto wiedzieć, że wszystko to, co da się obliczyć za pomocą komputerów, to funkcje budowane z kilku prostych cegiełek za pomocą prostych schematów: składania, operatora μ i rekurencji.

Załączam pozdrowienia dla Czytelników Pańskiego pisma

Matematyk

PS. Pozwolę sobie jeszcze zwrócić uwagę na pewien fakt wynikający z tego, co już napisałem. Wiadomo, że wszystkich liczb naturalnych jest nieskończenie wiele, ale jednak mniej niż liczb rzeczywistych. Możemy stwierdzić, po niezbyt skomplikowanych rozważaniach, że wszystkich funkcji rekurencyjnych jest dokładnie tyle, co liczb naturalnych. Z drugiej strony z twierdzeń teorii mnogości wynika, że wszystkich funkcji o argumentach i wartościach w zbiorze liczb naturalnych jest tyle, co wszystkich liczb rzeczywistych.

A więc to, co może zostać policzone przez najdoskonalsze nawet komputery, jest nieznaczną cząstką tego, co w ogóle możemy sobie wyobrazić!

JABŁKO WPADA DO WSPÓLNEGO KOSZYKA

Pod koniec ubiegłego roku prezes firmy Apple Computer John Sculley oświadczył, że firma i jej kooperanci w ciągu najbliższego roku rozpoczną dostawy na rynek urządzeń pozwalających na włączanie "jabłuszek" w systemy biurowe pracujące na standardzie IBM.

Tym samym firma Apple zapowiedziała zerwanie ze strategią, która była przedmiotem dumy Steve Jobsa, genialnego projektanta i jednego z założycieli przedsiębiorstwa, mianowicie utrzymywania niekompatybilności z IBM.

"Jabłuszka" były pierwszymi użytecznymi minikomputerami szkolnymi i domowymi dostarczanymi na rynek w ilościach handlowych, zanim gigant IBM zrobił to pod koniec lat siedemdziesiątych. Ale bardzo szybko koncentrując "atak" na tym segmencie rynku, na którym koncern tradycyjnie cieszył się największym poważaniem, mianowicie zastosowań biurowych, zdołał doprowadzić do zdynamizowania się całego rynku.

Obecnie udział IBM zbliża się do 50% rynku USA. Co piąty komputer osobisty – zaledwie! – kupuje osoba prywatna, a nie przedsiębiorstwo. Od paru już lat warunkiem przetrwania dla mniejszych firm jest kompatybilność z IBM PC. Apple Computer wskutek niekompatybilności traciła udział w rynku, potem zaczęła się zmniejszać sprzedaż, aż zmuszono do ustąpienia Jobsa ze stanowiska prezesa rady nadzorczej latem ub. r. usuwając w ten sposób przeszkody zawadzające przyjęciu bardziej handlowo skutecznej strategii firmy.

Musiała ona wreszcie przyłączyć się do kolumny, która wprawdzie wymaszerowała na jej hasło, ale w nieco innym kierunku.

(jal)

Strategia wygrywająca

Trudno powiedzieć, czy istnieje wyraźnie określona grupa "gier logicznych" i jakie właściwie gry można do niej zaliczyć. Tytuł ten wskazywać ma głównie na fakt, że nie będziemy zajmowali się tutaj grami zręcznościowymi, typu adventure czy losowymi. Co pozostaje? Duża grupa gier, które wymagają pewnej strategii, "logicznego myślenia" – jak się to czasami określa, nierzadko wiedzy itp. Na przykład kółko i krzyżyk, master mind, reversi, warcaby, szachy czy go. Że komputery potrafią grać w takie gry, o tym wiemy. Jeszcze przed masową produkcją mikrokomputera można było nabyć "elektronicznego partnera" do gry w szachy, a obecnie nie ma chyba właściciela komputera domowego, który nie posiadałby programu grającego w szachy czy warcaby.

Czy grając w jedną z tych gier z komputerem nie zdarzyło się nam zadziwić nad jego „inteligentnym zachowaniem"? Może poczuliśmy niepokój?

W cyklu tych kilku artykułów chciałbym powiedzieć na czym polega programowanie gier logicznych, przedstawić kilka przykładów, a także zachęcić do samodzielnych prób.

Zacznijmy od zastanowienia się na czym polega problem napisania programu grającego w taką grę. Wyobraźmy sobie, że wykonaliśmy ruch, kolej więc na akcję programu. W jaki sposób komputer może "wymyśleć" w miarę dobry ruch?

Nie ma jednej odpowiedzi na to pytanie, a to dlatego, że gra grze nierówna. Podstawowymi parametrami gier są: liczba wszystkich możliwych partii, inaczej określana liczbą kombinacji oraz to, czy znana jest czy nie strategia wygrywająca (lub nieprzegrywająca).

Co do drugiego problemu, to wiadomo, że strategia (jedna lub druga) istnieje (dla gier tu omawianych dział nauki z pogranicza matematyki i informatyki zwany teorią gier dostarcza stosownego dowodu). Natomiast parametr określający liczbę kombinacji wyraża złożoność gry. Gdy nie jest zbyt duży, tzn. taki, żeby zbadanie wszystkich możliwości stało się wykonalne w rozsądnym czasie, wówczas nie znając innej strategii można zbudować program oparty na tzw. przeszukiwaniu, czyli wybieraniu najlepszej z wszystkich możliwości. Przykład i dokładniejsze omówienie tej metody przedstawię innym razem.

Jednakże większość gier tak łatwo nie poddaje się zaprogramowaniu. Liczbę wszystkich możliwych partii w warcaby, szachy i go określają kolejne liczby: 10^{40} (10 do potęgi 40), 10^{120}, 10^{750}. Zakładając, że komputer jest w stanie zbadać milion ruchów w ciągu sekundy (co jest trochę za dużo nawet dla największych maszyn), potrzebowalibyśmy i tak około 3×10^{26} lat na jedną partię warcabów, co daje pojęcie o wielkości liczb, z jakimi mamy do czynienia. W porównaniu z tym, jakże przyjemna i prosta wydaje się perspektywa zaprogramowania gry, dla której znamy strategię wygrywającą. Wystarczy "przełożyć" ową strategię na język zrozumiały dla komputera i już. Jeżeli nawet nie zawsze jest to takie proste, to jednak prowadzi do dobrych wyników. Czasami nawet do zbyt dobrych – niektórym grom grozi zapomnienie tylko dlatego, że komputer radzi sobie z nimi zbyt łatwo.

Przykładem gry o znanej strategii jest NIM, częściej uprawiana, niż znana z nazwy. Najpowszechniej stosowanym rekwizytem w NIM są zapałki. Oto zasady. Gra dwóch graczy. Przed grą układają zapałki w trzy rzędy o dowolnej ilości w każdym. Ruch polega na zabraniu z jednego dowolnie wybranego rzędu dowolnej ilości zapałek – od jednej do wszystkich w rzędzie. Ruchy wykonują gracze na przemian. Przegrywa ten, który nie może wykonać ruchu, gdyż nie ma już zapałek w żadnym rzędzie.

Pod względem ilości kombinacji NIM jest grą tego samego typu co szachy i warcaby, szczególnie że ilość rzędów może być dowolnie duża. Jednakże istnienie strategii wygrywającej czyni z niej grę łatwą do zaprogramowania. Program taki gra w NIM nie popełniając błędów (nawet dla dużej liczby rzędów) i wygrywa zawsze, kiedy tylko jest to możliwe. Czy to znaczy, że strategia wygrywająca może czasami nie prowadzić do zwycięstwa? Nie, tak nie jest. Strategie wygrywające istnieją dla gier z tzw. pełną informacją, a w NIM ilość zapałek w rzędzie jest losowa. Dopiero po rozdzieleniu mamy pełną informację i możemy mówić o istnieniu strategii. Może się wówczas okazać, że polega ona między innymi na oddaniu pierwszego ruchu przeciwnikowi, gdyż kto zacznie, ten przegra. A porządny program zostawia człowiekowi decyzję, kto ma rozpocząć grę.

Ale dość, powiedzmy jaka jest ta strategia. Niech przykładowo będzie to 15 zapałek w pierwszym rzędzie, 8 w drugim i 11 w trzecim. Należy rozpocząć od zamiany liczb określających ilość zapałek w rzędzie z systemu dziesiętnego na dwójkowy. (Systemy liczbowe – patrz podręcznik do IV klasy szkoły podstawowej.) Oto mamy:

$15_{(10)} = 1111_{(2)}$
$8_{(10)} = 1000_{(2)}$
$11_{(10)} = 1011_{(2)}$

PROGRAMOWANIE

```
  10 REM NIM
  20 REM 27 to max ilość pionów
w jednym rzędzie, 6 minimalna
  30 LET ilrz =3: LET maxil=27:
LET minil=6: DIM b(3,5): DIM d(2
,5): DIM l(3,2)
  40 LET m=maxil-minil
  50 FOR k=1 TO ilrz
  60 LET il=minil+INT (m*RND)
  65 LET f=5*k
  70 FOR z=1 TO il
  80 PRINT AT f,z;"▌": PRINT AT
f+1,z;"▌"
  90 NEXT z
 100 LET l(k,1)=il
 110 NEXT k
 120 GO SUB 500
 130 INPUT "Czy mam zacząć? (t/n
) ";a$
 140 IF a$="n" OR a$="N" THEN GO
 TO 300
 200 GO TO 1000
 300 IF l(1,1)=0 AND l(2,1)=0 AN
D l(3,1)=0 THEN PRINT AT 10,5;"W
ygrałem !!!": FOR k=1 TO 7: BEEP
 .4,k+2: NEXT k: STOP
 310 INPUT "Podaj ruch:
             numer rzędu (1, 2 lu
b 3) ";i
 315 IF i<>1 AND i<>2 AND i<>3 T
HEN GO TO 310
 320 INPUT " ilość zabieranych
pałeczek ";il
 325 IF il<=0 OR il>l(i,1) THEN
GO TO 310
 330 GO SUB 3000
 350 GO TO 1000
```

Liczbę binarną otrzymujemy zapisując od prawej strony reszty z dzielenia liczby dziesiętnej przez 2 (patrz też list programu).
Sprawdźmy ostatni z otrzymanych wyników:

$1011_{(2)} = 1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 = 1 + 2 + 8 = 11$.

Dla tych, którzy niewiele wiedzą o systemach liczbowych i nie mają podręcznika do IV klasy – krótkie wyjaśnienie. Dlaczego w liczbie 125 wiadomo, że 1 oznacza 100 czyli 10^2? A gdyby stała tam cyfra 3, to czy wiedzielibyśmy, że oznacza 300 czyli 3×10^2? Zachodzi wzór ogólniejszy:
cyfra razy podstawa systemu do potęgi (n – 1), gdzie n jest pozycją cyfry w liczbie liczoną od prawej strony. Dlatego też 125 to 100 + 20 + 5, ale 125 w systemie ósemkowym to tylko 85 = 64 + 16 + 5.

Ustawiając binarne liczby jedna pod drugą otrzymujemy cztery kolumny zer i jedynek:

```
1 1 1 1
1 0 0 0
1 0 1 1
```

Strategia gry jest następująca:
Wykonuj takie ruchy, by liczba jedynek w każdej kolumnie była parzysta lub równa zeru. Jeżeli nie jest to możliwe przed wykonaniem pierwszego ruchu, zaproponuj rozpoczęcie gry przeciwnikowi.

W naszym przykładzie widzimy, że dwie pierwsze kolumny od lewej zawierają nieparzystą ilość jedynek. Należy ustalić, w którym rzędzie wykonać ruch oraz ile zabrać zapałek. Dlatego trzeba wybrać ten rząd, w którym występuje jedynka w najbardziej na lewo stojącej kolumnie o nieparzystej ilości jedynek – określmy ją jako kolumnę nieparzystą. W naszym przykładzie jest to kolumna pierwsza z lewej, a jedynka występuje w każdym z trzech rzędów. Można wybrać dowolny z nich. My weźmy pierwszy. Natomiast liczba zapałek, jaka ma pozostać w wybranym rzędzie jest liczbą, jaka powstaje po zamianie bitu (cyfra 0 lub 1) na przeciwny w kolumnach nieparzystych i pozostawieniu bez zmian w kolumnach parzystych. Stąd przez odjęcie obliczamy liczbę zapałek, które należy zabrać z rzędu.

W naszym przykładzie zamieniamy jedynki na zera w dwóch pierwszych kolumnach z lewej strony w rzędzie pierwszym. Daje to liczbę 0011 czyli 11, a w systemie dziesiętnym 3. Stąd poszukiwaną liczbą jest 15 – 3 = 12. Zauważmy, że odejmując 12 zapałek z pierwszego rzędu otrzymujemy parzyste liczby jedynek we wszystkich kolumnach. A o to przecież chodziło.

Pozostałe możliwe ruchy dla tego przykładu to zabranie z drugiego albo trzeciego rzędu czterech zapałek – zamiana 1 na 0 w kolumnach pierwszej i drugiej od lewej. Oba powyższe ruchy również dadzą same parzyste kolumny. Po każdym z nich przeciwnik musi przegrać, jeżeli tylko w dalszym ciągu kontynuować będziemy powyższą procedurę wyboru ruchu. Jak łatwo zgadnąć, nietrudno jest to zaprogramować. Tak przedstawia się program grający w NIM.

Warto może dodać jeszcze do niego kilka szczegółów, takich jak efekty dźwiękowe czy lepsza grafika. To jednak zostawiam dla wszystkich tych, którzy będą mieli na to ochotę. A tym, którzy zechcą zaprogramować własną grę radzę, by nie żałowali wysiłku przed rozpoczęciem programowania i zastanowili się nad strategią i algorytmem podczas szukania rozwiązań najlepszych i najprostszych. Może ktoś znajdzie strategię wygrywającą?

Janusz Kraszek

GIER LOGICZNYCH

```
500 >REM poczatek
505 FOR k=1 TO 5: LET d(1,k)=0:
NEXT k
510 FOR i=1 TO 3
520 GO SUB 2000
525 PRINT AT 5*i,30;l(i,1)
530 NEXT i
540 RETURN
1000 REM ruch
1010 LET rz=1: LET x=l(1,2): IF
x<l(2,2) THEN LET x=l(2,2): LET
rz=2
1020 IF x<l(3,2) THEN LET rz=3:
LET x=l(3,2)
1030 IF x=0 THEN PRINT AT 10,5;"
Przegrałem !?": BEEP .5,-7: BEEP
 .5,-9: STOP
1040 FOR k=x TO 1 STEP -1
1050 IF d(1,k)=1 THEN GO TO 1100
1060 NEXT k
1070 REM układ z parzystymi
         kolumnami
1080 LET il=1: LET i=rz: GO SUB
3000: GO TO 300
1100 FOR z=1 TO ilrz
1110 IF b(z,k)=1 THEN GO TO 1200
1120 NEXT z
1130 PRINT "bład - tu nigdy nie
powinniśmy się znaleźć": STOP
1200 LET l=0: LET p=1
1210 FOR w=1 TO l(z,2)
1220 LET c=b(z,w)
1230 IF d(1,w)=1 THEN LET c=1 AN
D b(z,w)=0
1240 LET l=l+p*c: LET p=2*p
1250 NEXT w
1260 LET il=l(z,1)-l: LET i=z: G
O SUB 3000: GO TO 300
```

```
2000 REM zapis dwojkowy
2005 FOR k=1 TO 5: LET d(2,k)=0:
 NEXT k
2010 LET x=l(i,1): LET j=0
2020 IF x=0 THEN GO TO 2060
2025 LET j=j+1: LET y=INT (x/2)
2030 LET c=x-2*y: LET d(2,j)=c
2050 LET x=y: GO TO 2020
2060 LET l(i,2)=j
2065 FOR k=1 TO 5
2070 IF d(2,k)<>b(i,k) THEN LET
d(1,k)=1 AND d(1,k)=0
2080 LET b(i,k)=d(2,k)
2090 NEXT k
2100 RETURN
3000 REM wykonaj ruch: rzad=i,
       ilosc to il
3300 LET l(i,1)=l(i,1)-il
3310 LET f=5*i
3400 FOR k=l(i,1)+1 TO 31
3500 PRINT AT f,k;" ": PRINT AT
f+1,k;" "
3600 NEXT k
3610 PRINT AT f,30;l(i,1)
3700 GO SUB 2000
3800 RETURN
```

GIEŁDA GIEŁDA

Podajemy notowania z giełdy mikrokomputerowej w Klubie Studenckim KARLIK w Krakowie z dnia 23 lutego 1986 r.:

ZX-81 + 16 K + klawiatura + dżojstik	45 tys.
Spectrum 16 K	65 tys.
Spectrum 48 K	90-95 tys.
Spectrum plus	120 tys.
Atari 600 XL	70 tys.
Atari 800 XL	100 tys.
magnetofon	40 tys.
Atari 130 XE	160 tys.
Commodore 64 + magnetofon	160 tys.
Commodore 64 + magn. + 2 dżojstiki	180 tys.
stacja dysków z interfejsem 1541	180 tys.
Commodore 116	ok. 60 tys.
Texas Instruments 99/4A	90 tys.
Sharp MZ700 + drukarka	powyżej 200 tys.
bez drukarki	150 tys.
MZ800	150 tys.
MSX (różnych producentów)	150 tys.
Amstrad 464, zielony monitor	350 tys.

(Pytał się i targował: Andrzej Załuski)

...oraz z giełdy w warszawskim klubie Stodoła z dnia 16 lutego 1986 r.:

Spectrum – w poszczególnych typach – ceny jak w Krakowie	
Commodore 64	150 tys.
dyskietki	1500-1800 sztuka
Interfejs 1 + Microdrive + kasetki (do Spectrum)	100 tys.
Amstrad 464, kolorowy monitor	400 tys.
Amstrad 464, zielony monitor	260 tys.

IBM PC składany z części z 640 K + drukarka + dyskietki	2.000.000

Kto ma dolary na koncie, może kupować w brytyjskich firmach wysyłkowych (niestety nie możemy podawać adresów, gdyż nikt nam za to nie zapłacił...):

Spectrum plus	95 funtów
QL	175 funtów
stacja dysków do Spectrum Discovery-Opus	1 napęd 160 f
	2 napędy 260 f
drukarka Seikosha CP50S	60 f
Interfejs 1 + Microdrive	90 f
ULA	8 f
Commodore 64	140 f
Commodore 128	235 f
stacja dysków 1570	180 f
Atari 130 XE	150 f
Atari 520 ST + monitor, dyski	650 f

BYTE z lutego 1986 r. informuje:

– firma ATARI wprowadziła na rynek nowy model 1040 ST. Wraz z wbudowanym napędem dyskowym i monitorem kosztuje on 1000 dol. Równocześnie cena Atari 520 ST (bez napędu dyskowego i monitora) została obniżona do 300 dol., przy czym model ten został skierowany do sprzedaży w supermarketach.

– Apple wprowadza na rynek nowy model Macintosh plus z 1 MB RAM i 128 kB ROM. Pamięć RAM może być rozszerzana do 4 MB.

– Chińska Republika Ludowa rozpoczęła masową licencyjną produkcję mikrokomputerów kompatybilnych z IBM PC. Można się z nimi porozumiewać zarówno po angielsku, jak i za pomocą hieroglifów.

KUP PAN CHIP, chip jest cheap – oto notowania z BYTE z lat 1983–1986 (ceny w dolarach):

	II.83	I.85	IV.85	X.85	II.86
mikroprocesor 8-bitowy Z-80A	5	3	3	2	1.70
pamięć statyczna SRAM 6116 150ns	5	5	3	2	1.60
mikroprocesor 16-bitowy 8088	40	20	20	9	7
koprocesor 8087	–	175	119	110	110
pamięć dynamiczna DRAM 4164 150ns (9 szt. po 64 kB)	63	45	22	6	6
DRAM 41256 150ns (1 szt. 256 kB)	–	–	–	–	2.50
EPROM 27128	30	19	13.5	3.5	2.80

W tym samym okresie CEMI nie wprowadziło żadnej znaczniejszej obniżki cen swych wyrobów.

(opr. WM)

GIEŁDA ■ GIEŁDA GIEŁDA ■ GIEŁDA ■ GIEŁDA

WKRÓTCE W KOMPUTERZE ■ WKRÓTCE W KOMPUTERZE

– o najnowszych propozycjach Amstrada: PCW8256 i PCW8512 oraz jego imporcie do Polski – o sieci lokalnej MERITUM i innych propozycjach krajowego przemysłu dla oświaty – o PTI – LOGO: gotowym już translatorze LOGO na Spectrum, znacznie ulepszonym w stosunku do LOGO SOLI – LCSI – o Pascalu i MS DOS – o straszliwej komputerowej pułapce dla żon i o tym, czy zdaniem dzieci komputery są żywe – o rodzinie mikroprocesorów firm MOTOROLA i MOSTEK, a w szczególności prawie wszystko o sercu COMMODORE – 6510 – o tym, jak zainstalować RESET w Spectrum i Commodore oraz o najprostszym banku danych – o zestawie programów POLONEZ i edytorze tekstu Tadeusza Wilczka
– o radzieckim programie edukacji informatycznej i o systemie oceny oprogramowania dydaktycznego stosowanym przez brytyjski Uniwersytet Otwarty
– dalsze programy edukacyjne: funkcje liniowe, chemia, cztery działania inaczej
– o konkursie RAZEM i telewizyjnego magazynu SPECTRUM
– algebraf, czyli program na komputerową krzyżówkę dla lubiących liczyć.